12-2017

Multi-user multi-keyword rank search over encrypted data in arbitrary language

Yang YANG
Fuzhou University

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1109/TDSC.2017.2787588

Published in IEEE Transactions on Dependable and Secure Computing , Dec 2017, Advance online
https://doi.org/0.1109/TDSC.2017.2787588

Multi-user Multi-Keyword Rank Search over Encrypted Data in Arbitrary Language

Yang Yang, *Member, IEEE,* Ximeng Liu, *Member, IEEE,* Robert H. Deng, *Fellow, IEEE,*

***Abstract*—Multi-keyword rank searchable encryption (MRSE) returns the top-k results in response to a data user's request of multi-keyword search over encrypted data, and hence provides an efficient way for preserving data privacy in cloud storage systems while without loss of data usability. Many existing MRSE systems are constructed based on an algorithm which we term as k-nearest neighbor for searchable encryption (KNN-SE). Unfortunately, KNN-SE has a number of shortcomings which limit its practical applications. In this paper, we propose a new MRSE system which overcomes almost all the defects of the KNN-SE based MRSE systems. Specifically, our new system does not require a predefined keyword set and supports keywords in arbitrary languages, is a multi-user system which supports flexible search authorization and time-controlled revocation, and it achieves better data privacy protection since even the cloud server is not able to tell which documents are the top-k results returned to a data user. We also conduct extensive experiments to demonstrate the efficiency of the new system.**

***Index Terms*—searchable encryption, multiple keyword, rank, top-k, privacy-preserving.**

I. Introduction

CLOUD computing [1] provides virtually unlimited computational and storage resources and has attracted increasing number of individuals and corporations to migrate their data into the cloud. The data privacy concerns the cloud computing brings along with it [2], [3] inspire cloud users to encrypt their sensitive documents before they are outsourced to cloud. Encryption translates data into unreadable ciphertext and how to search over and share encrypted data have been a challenging research problem. Searchable encryption (SE) [4], [5], [6] has been proposed as an effective method to execute keyword search over encrypted data. To make an encrypted document searchable in SE, a data owner firstly extracts a set of keywords from the document and encrypts them into an encrypted index. Then, the data owner uploads both the encrypted index and the encrypted document to the cloud for storage. In the data query phase, a data user creates a keyword token and submits the token to the cloud. The cloud uses a matching algorithm to test the association between the search token and encrypted indices. Then, the encrypted documents with matching keywords are returned to the data user.

Many efforts have been spent to study SE systems in diverse application scenarios, such as health care [7], [8], smart grid [9], internet of things [10]. Many of these SE systems only support single keyword search [11], [12] or simple conjunctive queries [13], [14], [15], [16], and are not able to rank searched documents according to some predefined relevance score.

To provide an improved search experience, the **m**ulti-keyword **r**ank **s**earchable **e**ncryption (MRSE) mechanism is proposed [28], [29], [30], [32] which allows the cloud to return the top-k results (with the k highest relevance scores) rather than all relevant documents. However, most of the existing MRSE systems are based on a special k-nearest neighbor (KNN) algorithm [28] which we will refer to as k-nearest neighbor algorithm for searchable encrytion (KNN-SE) during the rest of the paper. Unfortunately, as we will show in the paper, KNN-SE and hence the existing MRSE systems based on it suffer from many shortcomings which greatly limit their practical applications. It is necessary to design new MRSE systems to overcome these defects without loss of efficiency and security.

A. Related Work

The concept of secure search over encrypted data was firstly proposed by Song et al. [17] in 2000. Curmola et al. [18] proposed a searchable symmetric encryption (SSE) scheme to secure storage system. Cash et al. [19] suggested a scalable SSE scheme supporting boolean query. Liu et al. [20] utilized the SSE and attribute based encryption to securely share and search for real-time video data. Yang et al. [21] proposed a non-interactive order-preserving encryption scheme to search over encrypted database system, which supports range search. Zuo et al. [22] designed a SSE scheme to realize boolean search in secure database, which has higher efficiency compared with Cash's scheme [19]. Sun et al. [23] suggested a multi-client SSE system with support for boolean queries, which prevents the pre-query interaction between data owners and users. Kermanshahi et al. [24] also proposed a multi-user SSE scheme, which is constructed based on the oblivious cross tags protocol. Boneh et al. [25] put forth a public key encryption with keyword search scheme. Liang et al. [26] utilized the proxy re-encryption mechanism to design a public key searchable encryption scheme, and then they constructed a regular language search scheme over encrypted cloud data [27].

In 2011, Cao et al. [28] proposed the first framework of single-user MRSE based on KNN-SE. KNN-SE is a symmetric encryption system which utilizes "inner product similarity" to quantify the similarity and sort the result. A secret key in KNN-SE consists of two $k \times k$ matrices M_1, M_2 and a vector $S \in \{0,1\}^k$, where k is the number of predefined keywords in the system. For each file, the extracted keywords are mapped to a vector $I \in \{0,1\}^k$ such that each bit indicates whether a predefined keyword is in the file. Then, the vector I is split into two vectors I', I'' controlled by vector S. Finally, I', I'' is multiplied by $M_1^\top$ and $M_2^\top$, respectively, to generate an

encrypted index. The trapdoor generation is similar to the encrypted index except that the split query vector is multiplied with M_1^{-1} and M_2^{-1}. In the search phase, dot product is utilized to calculate the relevance score. (A brief overview of KNN-SE is given in Supplemental Materials A. The reader may refer to [28], [38] for more details.)

Since the publication of [28], most of the follow-on MRSE systems are constructed based on the KNN-SE approach. Yu et al. [29] put forth a two round searchable encryption system to realize top-k multi-keyword search. They employ KNN-SE (whcih is refereed to as the vector space model in [29]) and order preserving encryption techniques to improve the security.

Fu et al. [30] suggested a multi-keyword rank search system supporting synonym query based on KNN-SE. It enables synonym queries such that the possible synonym substitution can be tolerated in the data retrieval process. In [30], TF-IDF (term frequency-inverse document frequency) is leveraged in keyword extraction procedure. A data owner has to build an index tree to accelerate the search algorithm, which consumes more storage space. Later on, they proposed a verifiable keyword based semantic search system over encrypted data [31], which supports the verifiability of search result. They designed a symbol-based index tree to store the "path" information. The search result can be verified using this tree.

Sun et al. [32] also propose a verifiable searchable encryption system to support multi-keyword search and similarity-based ranking. They utilize tree-based index structure, multi-dimension algorithm and KNN-SE to improve the search efficiency. Li et al. [33] integrate the KNN-SE and blind storage methods to construct an MRSE with blind storage. Then, they [34], [35] utilize superincreasing sequence to construct a new MRSE system supporting boolean keyword query, such as "AND", "OR" and "NO" operations. They also leverage classified sub-dictionaries method to achieve better efficiency. Xia et al. [36] design a special tree-based index structure and a "greedy depth-first search" algorithm to enhance the search efficiency. They also used KNN-SE algorithm to encrypt the index and query.

Chen et al. [37] construct a hierarchical clustering method to enable more search semantics. Based on the minimum relevance threshold, the hierarchical method clusters encrypted documents and partitions the resulting clusters into sub-clusters. In that way, they achieve better search speed. Fu et al. [38] use locality sensitive hash function, Bloom filter and KNN-SE to realize a multi-keyword fuzzy searchable encryption system.

Although a lot of the MRSE systems are built based on the KNN-SE algorithm, the algorithm actually has several obvious limitations. Firstly, KNN-SE requires a predefined keyword set at the system setup phase and the entire system need to be rebuilt in order to add any new keywords into the system. Secondly, KNN-SE is a symmetric key encryption system and hence a data owner has to disclose his secret key to a data user in order to authorize the latter query and decryption privileges and the authorization can not be revoked even if the authorized data user is found behaving maliciously. Thirdly, it's impossible for KNN-SE based MRSE systems to support keyword search in arbitrary languages since the number of keywords and hence the dimensions of the matrices M_1 and M_2 will be astronomical in order to supports all keywords in all languages. Fourth, the relevance scores of the documents are in plaintext and the cloud server could learn the statistic information of data, such as which are the high relevant documents and the high-frequency returned files. These information will leak user's privacy.

This brief defects analysis of KNN-SE brings us a simple conclusion: existing MRSE systems based on KNN-SE are not suitable for practical applications.

B. Contributions

In this paper we propose a new MRSE system which overcomes all the limitations of the KNN-SE based MRSE systems while without loss of efficiency and security. In particular, our new MRSE enjoys the following desirable properties.

- ***No need for pre-defined keywords***. The new system does not require a set of pre-define keywords during the setup phase and new keywords can be added any time during the system operation.
- ***Support arbitrary language***. We use Unicode [48] to encode keywords in arbitrary languages and utilize an efficient way to transform them into encrypted ciphertext.
- ***Flexible authorization and time-controlled revocation***. The system allows a data owner to authorize a data user the search and decryption privileges in a specified period of time. When the current time is out of the defined time period, the right will be revoked automatically. Moreover, the system also provides the data owner an effective way to deprive the authorized privileges within a time period.
- ***Simultaneously search on multi-owner's data***. A data user can simultaneously search on multiple data owner's encrypted documents. For instance, a medical doctor can simultaneously search over encrypted medical records produced by different clinics. The other existing searchable encryption schemes have to generate different trapdoors to search over different data owner's documents. While, our scheme could use only one trapdoor to search on multiple owners' data.
- ***Flexible keyword weight and preference setting***. In the encryption phase, a data owner can set different keyword weights according to the importance of these keywords. In the query phase, a data user can search on multiple keywords and set different preference scores for each keyword. In the search phase, a cloud server can compute the relevance scores in encrypted form according to the keyword weight and preference scores and returns top-k results to the data user.
- ***Security and Efficiency***. In the existing MRSE systems, the server is able to learn the plaintext of relevance scores of each searched document. A cloud server lerans which documents are the most relevant. In our system, the cloud server learns nothing from the search results since the relevance scores returned to data user are encrypted. We prove the security of this system and conduct extensive simulations to demonstrate its efficiency.

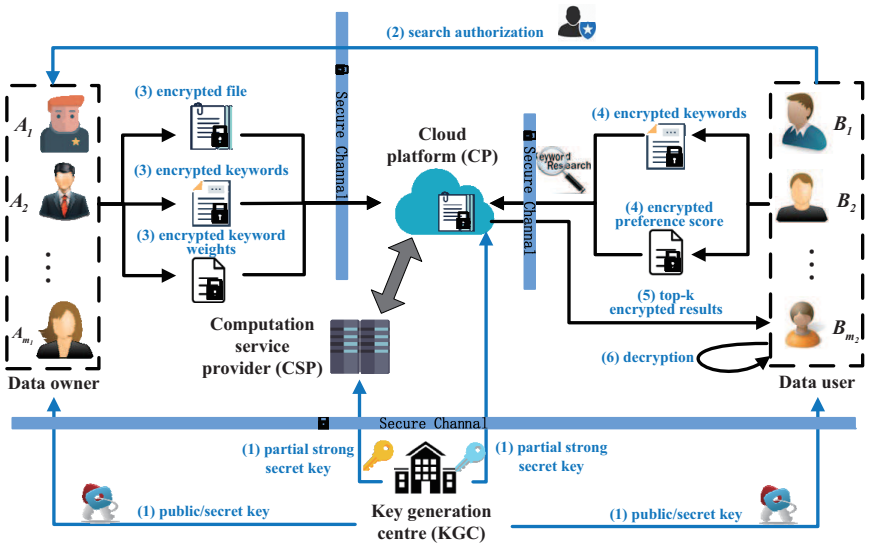


Fig. 1: System Model

II. System Architecture and Security Model

A. System Model

The system shown in Figure 1 consists of five entities.

- ***Key generation center (KGC)*** is trusted by all entities in the system, who is responsible to generate keys for each entities in the system.
- ***Cloud platform (CP)*** has powerful storage and computation abilities and stores user's files in an encrypted form. CP also responds on users' computation and data retrieval requests.
- ***Computing service provider (CSP)*** is an online computation server and processes strong calculational capabilities. It executes interactive computations with CP.
- ***Data owner***. Data owner encrypts his documents and outsources them to CP.
- ***Data user***. Data user generates a keyword trapdoor to issue data retrieval request to CP.

B. Attack Model

We follow the attack model in [42], [43], in which KGC is a fully trusted entity, and CP and CSP are "honest-but-curious" who are honest to execute the protocols but curious with the plaintext of user data. An adversary $\mathcal{A}^*$ is defined in this attack model, whose purpose is to recover the plaintext of data owner's privacy-preserving documents and data user's retrieved results. $\mathcal{A}^*$ has the following abilities.

(1) $\mathcal{A}^*$ could *eavesdrop* all communications to get the transmitted information.
(2) $\mathcal{A}^*$ could *compromise* CP and try to get the plaintext from the encrypted information sent by the data owner and CSP.
(3) $\mathcal{A}^*$ could *compromise* CSP and try to obtain the plaintext from the ciphertext sent by CP in interactive protocol.
(4) $\mathcal{A}^*$ could *compromise* a set of data users (except the challenge user) and get their privileges. $\mathcal{A}^*$ wants to get the plaintext information that belong to the challenge user.

However, the attacker $\mathcal{A}^*$ is not allowed to compromise: (1) CP and CSP simultaneously, (2) the challenge user. These are typical restrictions in cryptographic protocols [44].

C. Security Model

The security model in [45], [46] is adopted in this work. Consider three parties: system user (a.k.a "D_1"), CP (a.k.a "S_1") and CSP (a.k.a "S_2"), where system user includes data owner and user. We construct three simulators $(Sim_{D_1}, Sim_{S_1}, Sim_{S_2})$ against three types of attackers $(\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ that corrupt D_1, S_1 and S_2, respectively. These attackers are deemed as non-colluding and semi-honest. Due to the length limitation, please refer to [40], [45], [46] for the general security model definitions.

III. Crypto Primitives and Protocols

A. Paillier Cryptosystem with Threshold Decryption

The Paillier cryptosystem [39] with threshold decryption (PCTD) in [40], [41] is utilized for data encryption and could prevent the private key leakage risk in this paper. Let $\mathcal{L}(X)$ denote the bit length of X.

KeyGen: Let k be the security parameter and p, q be two large prime numbers such that $\mathcal{L}(p) = \mathcal{L}(q) = k$. Let $N = pq$ and $\lambda = lcm(p-1, q-1)/2$ [1]. Define a function $L(x) = \frac{x-1}{N}$ and select a generator g of order $ord(g) = (p-1)(q-1)/2$. The system public parameter is $PP = (g, N)$. The master secret key of the system is $SK = \lambda$. A user i in the system is assigned a secret key $sk_i \in Z_N$ and a public key $pk_i = g^{sk_i} \bmod N^2$.

Encryption: On input a plaintext $m \in \mathbb{Z}_N$, a user randomly selects $r \in [1, N/4]$ and uses his public key pk_i to encrypt m

[1] lcm : lowest common multiple.

to ciphertext $[m]_{pk_i} = (C_1, C_2)$, in which $C_1 = pk_i^r(1+mN) \mod N^2$ and $C_2 = g^r \mod N^2$.

Decryption with weak secret key: On input ciphertext $[m]_{pk_i}$ and weak private key sk_i, the message can be recovered by computing $m = L(C_1/C_2^{sk_i} \mod N^2)$.

Decryption with master secret key: Using master secret key $SK = \lambda$ of the system, any ciphertext $[m]_{pk_i}$ encrypted by any public key can be decrypted by computing $C_1^{\lambda} = (pk_i^r)^{\lambda}(1 + mN\lambda) = (1 + mN\lambda) \mod N^2$. Since $gcd(\lambda, N) = 1$ holds [2], we have $m = L(C_1^{\lambda} \mod N^2)\lambda^{-1} \mod N$.

Master secret key splitting: The master secret key $SK = \lambda$ can be randomly split into two parts $SK_1 = \lambda_1$ and $SK_2 = \lambda_2$ such that $\lambda_1 + \lambda_2 \equiv 0 \mod \lambda$ and $\lambda_1 + \lambda_2 \equiv 1 \mod N^2$.

Partial Decryption with SK_1 (`PD1`): On input the ciphertext $[m]_{pk_i} = (C_1, C_2)$, we can use $SK_1 = \lambda_1$ to compute $C_1^{(1)} = (C_1)^{\lambda_1} = (pk_i^r)^{\lambda_1}(1 + mN\lambda_1) \mod N^2$.

Partial Decryption with SK_2 (`PD2`): On input $[m]_{pk_i}$ and $C_1^{(1)}$, we can use $SK_2 = \lambda_2$ to compute $C_1^{(2)} = (C_1)^{\lambda_2} = (pk_i^r)^{\lambda_2}(1+mN\lambda_1) \mod N^2$. The message can be recovered by computing $m = L(C_1^{(1)} \cdot C_1^{(2)})$.

Ciphertext Refresh (`CR`): `CR` algorithm is utilized to refresh a ciphertext $[m]_{pk_i} = (C_1, C_2)$ into a new ciphertext $[m']_{pk_i} = (C_1', C_2')$ such that $m = m'$. It selects a random $r' \in \mathbb{Z}_N$, calculates $C_1' = C_1 \cdot pk_i^{r'} \mod N^2$ and $C_2' = C_2 \cdot g^{r'} \mod N^2$.

It is easy to verify that PCTD is **additive homomorphic**: $[m_1]_{pk_i} \cdot [m_2]_{pk_i} = [m_1 + m_2]_{pk_i}$ and $([m]_{pk_i})^r = [r \cdot m]_{pk_i}$ for a random $r \in Z_N$.

The following two protocols in [40] are also utilized in the proposed system. Let pk_A and pk_B be the public keys of users A and B. pk_Σ is another public key that will be defined later.

Secure Addition Protocol across Domains (`SAD`): Given $[X]_{pk_A}$ and $[Y]_{pk_B}$, `SAD` protocol securely calculates $[X + Y]_{pk_\Sigma}$.

Secure Multiplication Protocol across Domains (`SMD`): Given $[X]_{pk_A}$ and $[Y]_{pk_B}$, `SMD` protocol securely calculates $[X \cdot Y]_{pk_\Sigma}$.

B. Keyword Representation and Encryption

An important issue is how can we transform a keyword into a ciphertext to support any keyword in any language without predefined keyword set. We design a secure keyword to ciphertext algorithm (`K2C`) to fulfill the requirement, which mainly includes the following steps. (1) Map each character (including the special character) in the keyword into its unicode (UTF-16: 16-bit Unicode Transformation Format) [48]; (2) convert each hexadecimal unicode into decimal integer; (3) according to the position of the character in the keyword, multiply the decimal integer of the character with a weight; (4) add all the weighted decimal integer into a big integer; and (5) utilize the PCTD encryption algorithm and data owner's public key to encrypt the big integer of the keyword into a ciphertext.

An example is given in Fig. 2 to illustrate how to transform the string "keyword" in English, Chinese, Korean and Japanese languages into a ciphertext using K2C algorithm. It is worth noting that the K2C algorithm can successfully convert a keyword into a unique big integer, which can greatly solve the erroneous probability problem by using the bloom filter in other searchable encryption schemes [38], [47].

C. Encrypted Keyword Equivalence Testing Protocol

After keyword is encrypted, another crucial issue is to test whether two keyword ciphertexts contain the same keyword. A secure keyword equivalent test protocol across domains (`KET`) is designed to fulfill the task. Input two encrypted keywords $[X]_{pk_A}$ and $[Y]_{pk_B}$ that are encrypted by different public keys, `KET` protocol outputs an encrypted result $[u^*]_{pk_\Sigma}$ to indicate whether two keywords are the same. In order to make the protocol properly work, we require that $\mathcal{L}(X), \mathcal{L}(Y) < \mathcal{L}(N)/4$. CP and CSP will jointly execute the protocol using the assigned keys SK_1 and SK_2, respectively.

Step 1: CP computes

$$
\begin{aligned}
[X_1]_{pk_A} &= ([X]_{pk_A})^2 \cdot [1]_{pk_A} = [2X + 1]_{pk_A};\\
[Y_1]_{pk_B} &= ([Y]_{pk_B})^2 = [2Y]_{pk_B};\\
[X_2]_{pk_A} &= ([X]_{pk_A})^2 = [2X]_{pk_A};\\
[Y_2]_{pk_B} &= ([Y]_{pk_B})^2 \cdot [1]_{pk_B} = [2Y + 1]_{pk_B}.
\end{aligned}
$$

Then, CP randomly selects r_1, r_2, r_3, r_4 such that $\mathcal{L}(r_1), \mathcal{L}(r_2) < \mathcal{L}(N)/4 - 1$, $\mathcal{L}(r_3), \mathcal{L}(r_4) < \mathcal{L}(N)/8$. Then, CP flips random coins $s_1, s_2 \in \{0, 1\}$.

CP and CSP jointly execute the following operations.

If $s_1 = 1$, $[\gamma_1]_{pk_\Sigma} = \texttt{SAD}(([X_1]_{pk_A})^{r_1}, ([Y_1]_{pk_B})^{N-r_1})$.
If $s_1 = 0$, $[\gamma_1]_{pk_\Sigma} = \texttt{SAD}(([X_1]_{pk_A})^{N-r_1}, ([Y_1]_{pk_B})^{r_1})$.
If $s_2 = 1$, $[\gamma_2]_{pk_\Sigma} = \texttt{SAD}(([X_2]_{pk_A})^{N-r_2}, ([Y_2]_{pk_B})^{r_2})$.
If $s_2 = 0$, $[\gamma_2]_{pk_\Sigma} = \texttt{SAD}(([X_2]_{pk_A})^{r_2}, ([Y_2]_{pk_B})^{N-r_2})$.

CP calculates $l_1 = [\gamma_1]_{pk_\Sigma} \cdot [r_3]_{pk_\Sigma}$, $l_2 = [\gamma_2]_{pk_\Sigma} \cdot [r_4]_{pk_\Sigma}$, $l_1' = \texttt{PD1}_{SK_1}(l_1)$, $l_2' = \texttt{PD1}_{SK_1}(l_2)$ and sends (l_1, l_1', l_2, l_2') to CSP.

Step 2: CSP decrypts $l_1'' = \texttt{PD2}_{SK_2}(l_1, l_1')$, $l_2'' = \texttt{PD2}_{SK_2}(l_2, l_2')$. If $\mathcal{L}(l_1'') > \mathcal{L}(N)/2$, CSP sets $u_1' = 0$ and $u_1' = 1$ otherwise. If $\mathcal{L}(l_2'') > \mathcal{L}(N)/2$, CSP sets $u_2' = 0$ and $u_2' = 1$ otherwise. Then, CSP encrypts u_1', u_2' (with public key pk_Σ) to $([u_1']_{pk_\Sigma}, [u_2']_{pk_\Sigma})$, which are sent to CP.

Step 3: Receiving $([u_1']_{pk_\Sigma}, [u_2']_{pk_\Sigma})$, CP calculates as below.

If $s_1 = 1$, CP computes $[u_1]_{pk_\Sigma} = [u_1']_{pk_\Sigma}$; otherwise, CP computes $[u_1]_{pk_\Sigma} = [1]_{pk_\Sigma} \cdot ([u_1']_{pk_\Sigma})^{N-1} = [1 - u_1']_{pk_\Sigma}$.

If $s_2 = 1$, CP computes $[u_2]_{pk_\Sigma} = [u_2']_{pk_\Sigma}$; otherwise, CP computes $[u_2]_{pk_\Sigma} = [1]_{pk_\Sigma} \cdot ([u_2']_{pk_\Sigma})^{N-1} = [1 - u_2']_{pk_\Sigma}$.[3]

Then, CP and CSP jointly calculates $[u^*]_{pk_\Sigma} = \texttt{SMD}([u_1]_{pk_\Sigma}, [u_2]_{pk_\Sigma})$.

If $u^* = 1$, it indicates that the two keywords are the same. Otherwise, $u^* = 0$.

[2] *gcd*: greatest common divider.

[3] If $u_1 = 1$, it indicates $X \geq Y$; otherwise, $u_1 = 0$. If $u_2 = 1$, it indicates $Y \geq X$; otherwise, $u_2 = 0$.

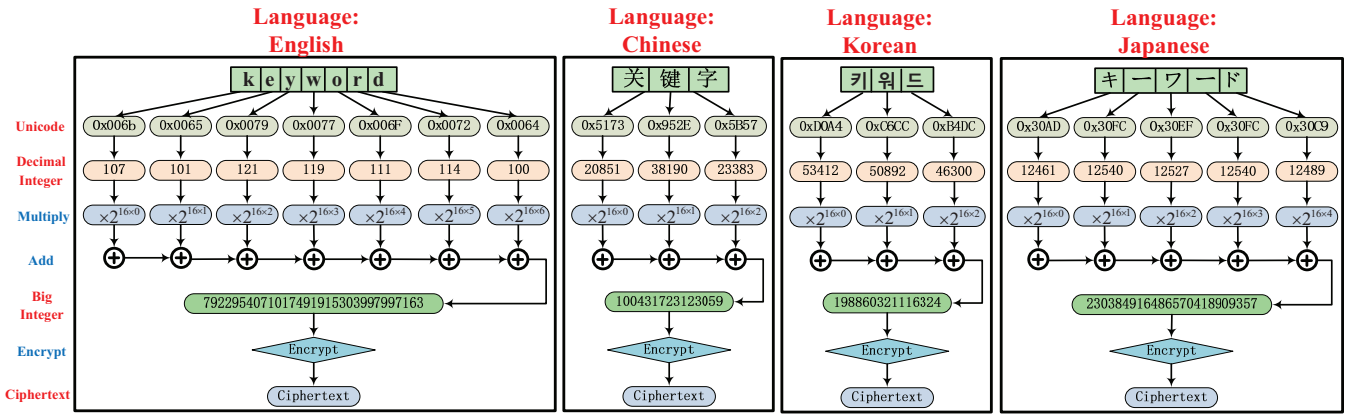


Fig. 2: Example of Keyword to Ciphertext

IV. Proposed System

In the proposed system (shown in Fig. 1), KGC firstly generates the master secret key for the system and the public/secret key pairs for the users, and partial strong secret keys for the CP and CSP (see step 1 in Fig. 1). Then, the data users apply search authorization from a single data owner or multiple data owners (step 2). The data owner sets an authorization time such that the search privilege expires automatically when the time is out. In the encryption phase (step 3), the data owner encrypts the file and a set of keywords, which are extracted from the file. To measure the importance of the keywords, a set of keyword weights are set and encrypted by the data owner. The encrypted files and keyword indexes are then uploaded to the CP. In the query phase (step 4), the data user designates a set of query keywords and a set of preference scores for the query keywords. Then, he generates the trapdoor of the query keywords and preference scores. The trapdoor is sent to the CP to issue a search query. In the search phase, the CP verifies the search authority of the data user. If it is valid, CP calculates the relevance score of the encrypted file and the trapdoor. The calculated relevance score is in the encrypted form. Then, CP leverages the secure data retrieval protocol to get the top-k relevant encrypted files, which are returned to the data user (step 5). In the decryption phase, the data user recovers the top-k plaintext file (step 6). The communication channel in the system should be secure channel, protected using, for example, SSL (Secure Sockets Layer) or TLS (Transport Layer Security). Table I shows the main notations in this paper.

TABLE I: Notations

Notation	Description
MPK/MSK	master public/secret key
pk/sk	public/secret key of user
SK_1, SK_2	partial strong secret keys
$\mathcal{L}(X)$	bit length of X
$[X]_{pk}$	the ciphertext of X (encrypted by pk)
$SEnc/SDec$	symmetric encryption/decryption algorithm
$Sig/Verify$	signature/verification algorithm
CER/RVK	authorization/revocation certificate
AT/RT	authorization time period/revocation time
kw/α	extracted keyword/keyword weight
qw/β	query keyword/preference score
$\mathbb{W}/\mathbb{Q}$	encrypted index/encrypted query
$M/ID/K$	file/file identity/file encryption key
$[I]_{pk_\Sigma}$	encrypted relevance score
K2C	secure keyword to ciphertext algorithm
KET	secure keyword equivalent test protocol across domains
MKS	secure multiple keyword search protocol across domains
MAX	secure maximum protocol across domains
MAX$_n$	secure maximum out of n protocol across domains
Top-K	secure top-k data retrieval protocol across domains

A. Key Generation

KGC runs $KeyGen$ algorithm of PCTD (Section III) to generate the system public parameter $PP = (g, N)$, strong master secret key $MSK = \lambda$ and user A_i's public/secret key pair $pk_{A_i} = g^{\theta_i}, sk_{A_i} = \theta_i$. KGC also generates a master public key $MPK = g^{\lambda}$. MSK is kept secret by KGC and MPK is public. Then, KGC executes master secret key splitting algorithm of PCTD to generate partial strong secret keys $SK_1 = \lambda_1$ and $SK_2 = \lambda_2$, which are secretly sent to CP and CSP, respectively. sk_{A_i} is confidentially sent to user A_i and pk_{A_i} is public.

Let $SEnc/SDec$ denote a symmetric encryption/decryption pair (with key space $\mathcal{K}$) that is cryptographically secure. Let $Sig/Verify$ denote signature generation/verification pair that is strongly unforgeable. The concrete algorithms will not be specified in this paper. We also define hash functions $H_1 : \{0,1\}^* \rightarrow Z_N$ and $H_2 : Z_N \rightarrow \mathcal{K}$.

To simplify the presentation, we utilize the element in Z_N to be the secret key of Sig algorithm. In practical usage, the signature key can be easily derived from the element in Z_N using a hash function.

B. User Authorization and Revocation

1) Single Data Owner Scenario: Suppose user B wants to be authorized to search over data owner A_1's data from 1st Jan. 2016 to 1st Jan. 2017 (authorization time AT_1 = "20160101 − 20170101"). User B should send the message (B, AT_1) to data owner A_1 to apply for the authorization. If it is permitted, A_1 will generate the authorization certificate for B.

$$CER_{A_1,B} = \langle cer = (A_1, B, AT_1, pk_\Sigma), Sig(cer, sk_{A_1}) \rangle,$$

where $pk_\Sigma = g^{sk_\Sigma}, sk_\Sigma = H_1(A_1, B, AT_1, sk_{A_1})$.

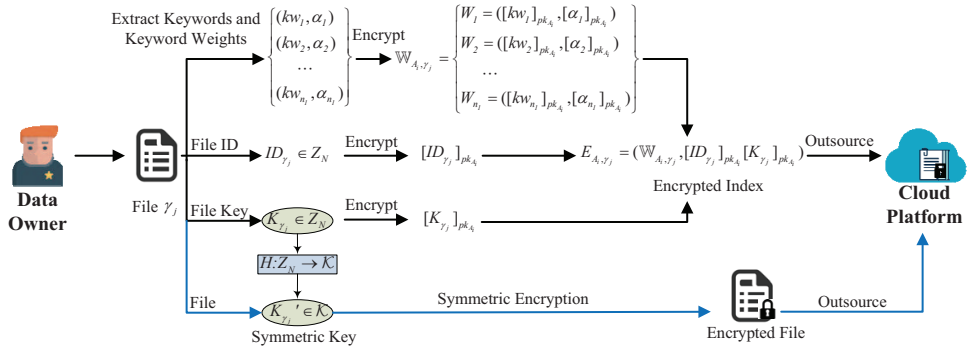


Fig. 3: Encryption Illustration

The secret key sk_Σ is given to B via secure channel. $CER_{A_1,B}$ will be sent to KGC, CP, CSP and user B. It will automatically be invalid when the current time is out of AT_1 and user's authorization will be revoked.

If A_1 plans to revoke B's privilege in the period of AT_1, he should create a revocation certificate $RVK_{A_1,B}$ as

$$\langle rvk = (CER_{A_1,B}, revoke, RT), Sig(rvk, sk_{A_1})\rangle,$$

where RT is the revocation time. Then, $RVK_{A_1,B}$ is sent to KGC, CP, CSP and user B.

2) Multiple Data Owners Scenario: Suppose user B wants to simultaneous query information from data owners $(A_1, \cdots, A_m)$'s files. He should firstly get the permissions $(CER_{A_i,B}, 1 \le i \le m)$ from each owner. Then, he applies the simultaneously query authorization from KGC. After verifying the certificates, KGC will calculate the authorization time period $AT_\Sigma = AT_1 \bigcap \cdots \bigcap AT_m$. Then KGC creates authorization certificate $CER_{\Sigma,B}$ as

$$\langle cer = (A_1, \cdots, A_m, B, AT_\Sigma, pk_\Sigma), Sig(cer, MSK)\rangle,$$

where $pk_\Sigma = g^{sk_\Sigma}, sk_\Sigma = H_1(A_1, \cdots, A_m, B, AT_\Sigma, MSK)$. sk_Σ is confidentially delivered to user B and pk_Σ is public to CP, CSP and user B.

To revoke $CER_{A_i,B}$ within the time period AT_Σ, KGC has to generate a revocation certificate $RVK_{\Sigma,B}$ as

$$\langle rvk = (CER_{\Sigma,B}, revoke, RT), Sig(rvk, MSK)\rangle,$$

where RT is the revocation time. Then, $RVK_{\Sigma,B}$ is sent to CP, CSP and user B.

C. Encryption

Suppose a data owner A_i want to upload a file M_{γ_j} to the cloud. He follows the steps below to encrypt the data. Fig. 3 illustrates the encryption algorithm.

1) Data owner A_i extracts a set of keywords $\{kw_1, \cdots, kw_{n_1}\}$ to describe the file. In order to differentiate the importance of the keyword, A_i sets keyword weights $\{\alpha_1, \cdots, \alpha_{n_1}\} \in Z_N$ for the keywords. There are many ways to compute keyword weight, such as TF-IDF (term frequency and inverse document frequency). The data owner selects a method to define the keyword weight, which is not specified in this paper.
2) A_i encrypts the keywords using **K2C** algorithm to get $\{[kw_1]_{pk_{A_i}}, \cdots, [kw_{n_1}]_{pk_{A_i}}\}$. The keyword weights are encrypted (using the PCTD algorithm) to $\{[\alpha_1]_{pk_{A_i}}, \cdots, [\alpha_{n_1}]_{pk_{A_i}}\}$. A pair of encrypted keyword and keyword weight is denoted as $W_k = ([kw_k]_{pk_{A_i}}, [\alpha_k]_{pk_{A_i}})$, $1 \le k \le n_1$. Denote $\mathbb{W}_{A_i,\gamma_j} = (W_1, \cdots, W_{n_1})$.
3) The file identity $ID_{\gamma_j} \in Z_N$ and file encryption key $K_{\gamma_j} \in Z_N$ are encrypted (using the PCTD algorithm) to $[ID_{\gamma_j}]_{pk_{A_i}}$ and $[K_{\gamma_j}]_{pk_{A_i}}$.
4) Using hash function $H_2 : Z_N \to \mathcal{K}$, the file encryption key K_{γ_j} is hashed to $K'_{\gamma_j} = H_2(K_{\gamma_j}) \in \mathcal{K}$. Then, A_i utilizes symmetric encryption algorithm $SEnc$ to encrypt the file M_{γ_j} into an encrypted file $C_{\gamma_j} = SEnc(M_{\gamma_j}, K'_{\gamma_j})$.
5) A_i sends $E_{A_i,\gamma_j} = (\mathbb{W}_{A_i,\gamma_j}, [ID_{\gamma_j}]_{pk_{A_i}}, [K_{\gamma_j}]_{pk_{A_i}})$ and encrypted file C_{γ_j} to cloud platform.

Discussion: If the keyword weight is a decimal (such as TF-IDF), the data owner could multiply all the keyword weights with an integer (such as 10 or 100) such that these decimals are mapped into Z_N.

D. Query

In the query phase, user B generates a trapdoor to issue the query (Fig. 4).

1) B specifies the query keywords $\{qw_1, \cdots, qw_{n_2}\}$ and the preference scores of the keywords $\{\beta_1, \cdots, \beta_{n_2}\}$, which indicates the importance of the keyword in the query.
2) B uses **K2C** to encrypt the query keywords and obtains $\{[qw_1]_{pk_B}, \cdots, [qw_{n_2}]_{pk_B}\}$. The preference scores are encrypted (using the PCTD algorithm) to $\{[\beta_1]_{pk_B}, \cdots, [\beta_{n_2}]_{pk_B}\}$. Denote tuple $Q_k = \{[qw_k]_{pk_B}, [\beta_k]_{pk_B}\}$ $(1 \le k \le n_2)$ and $\mathbb{Q} = (Q_1, \cdots, Q_{n_2})$.
3) B signs on the query $\mathbb{Q}$ using his private key sk_B and generates a signature $S_\mathbb{Q} = Sig(\mathbb{Q}, sk_B)$.
4) B sends the encrypted query $\mathbb{Q}$, signature $S_\mathbb{Q}$ and his identity User_B to cloud platform.

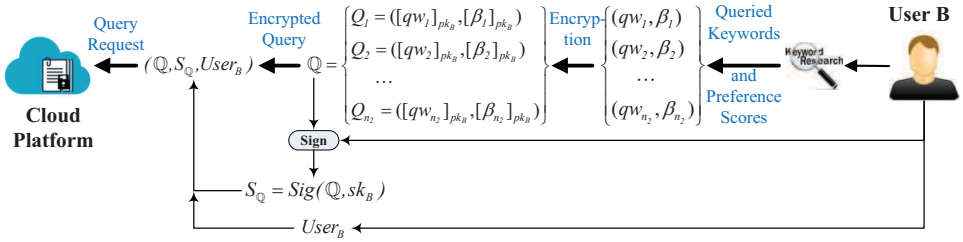


Fig. 4: Query Illustration

E. Search

Receiving the keyword search request, cloud platform firstly verifies whether B is authorized to access to the data. If B has the privilege, CS verifies the signature $S_{\mathbb{Q}}$ of encrypted query $\mathbb{Q}$ using user B's public key pk_B. If it is not valid, the query will be rejected. Otherwise, CP responds to the search query. (1) CP will firstly compute the relevance score of the encrypted query $\mathbb{Q}$ and the encrypted index for each authorized document. (2) Then, the encrypted files that have the top-k highest relevance scores will be returned to user. The concrete procedures are illustrated as below.

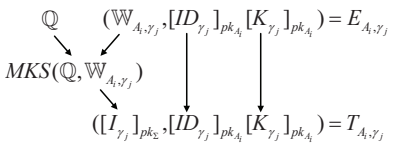


Fig. 5: Relevance Score Computation

E.1. Relevance Score Computation

In order to calculate the relevance score between search query and the file index, we design a secure multiple keyword search protocol across domains (**MKS**) to compute the score.

Algorithm 1: SECURE MULTIPLE KEYWORD SEARCH PROTOCOL ACROSS DOMAINS (**MKS**)

Input: Encrypted indexes $\mathbb{W} = (W_1, \cdots, W_{n_1})$ and search query $\mathbb{Q} = (Q_1, \cdots, Q_{n_2})$, where $W_i = \langle [X_i]_{pk_A}, [\alpha_i]_{pk_A} \rangle$, $Q_j = \langle [Y_j]_{pk_B}, [\beta_j]_{pk_B} \rangle$.
Output: $[I]_{pk_\Sigma}$.

1 Initialize $[I]_{pk_\Sigma} = [0]_{pk_\Sigma}$;
2 **for** $j = 1$ *to* n_2 **do**
3 **for** $i = 1$ *to* n_1 **do**
4 CP and CSP jointly calculate $[u_i]_{pk_\Sigma} = \mathbf{KET}([X_i]_{pk_A}, [Y_j]_{pk_B})$;
5 $[s'_i]_{pk_\Sigma} = \mathbf{SMD}([\alpha_i]_{pk_A}, [\beta_j]_{pk_B})$;
6 $[s_i]_{pk_\Sigma} = \mathbf{SMD}([u_i]_{pk_\Sigma}, [s'_i]_{pk_\Sigma})$;
7 CP calculates $[I]_{pk_\Sigma} = [I]_{pk_\Sigma} \cdot [s_i]_{pk_\Sigma}$;
8 **Return** $[I]_{pk_\Sigma}$.

The inputs of **MKS** protocol are the encrypted index $\mathbb{W}_{A_i,\gamma_j} = (W_1, \cdots, W_{n_1})$ and the encrypted query $\mathbb{Q} = (Q_1, \cdots, Q_{n_2})$, where $W_{k_1} = ([kw_{k_1}]_{pk_{A_i}}, [\alpha_{k_1}]_{pk_{A_i}})$ for $1 \le k_1 \le n_1$ and $Q_{k_2} = \{[qw_{k_2}]_{pk_B}, [\beta_{k_2}]_{pk_B}\}$ for $1 \le k_2 \le n_2$. It will output an encrypted relevance score $[I]_{pk_\Sigma}$ under public key pk_Σ.

For each queried keyword Y_j ($1 \le j \le n_2$ in Line 2), the **MKS** protocol calculates its relevance score with the encrypted index. In order to achieve this purpose, the algorithm firstly computes the relevance score of Y_j and X_i ($1 \le i \le n_1$ in Line 3).

1) In line 4, it utilizes **KET** algorithm to test whether $X_i = Y_j$. It outputs $[u_i]_{pk_\Sigma} = [1]_{pk_\Sigma}$ if $X_i = Y_j$. Otherwise, it outputs $[u_i]_{pk_\Sigma} = [0]_{pk_\Sigma}$.
2) In line 5, it calculates the multiplication of the keyword weight α_i (of X_i) and the preference score β_j (of Y_j)
$$[s'_i]_{pk_\Sigma} = [\alpha_i \cdot \beta_j]_{pk_\Sigma} = \mathbf{SMD}([\alpha_i]_{pk_A}, [\beta_j]_{pk_B}).$$
3) In line 6, if $X_i = Y_j$, the relevance score of X_i and Y_j is $[s_i]_{pk_\Sigma} = [\alpha_i \cdot \beta_j]_{pk_\Sigma}$ since
$$\mathbf{SMD}([u_i]_{pk_\Sigma}, [s'_i]_{pk_\Sigma}) = \mathbf{SMD}([1]_{pk_\Sigma}, [\alpha_i \cdot \beta_j]_{pk_\Sigma}).$$
Otherwise, $[s_i]_{pk_\Sigma} = [0]_{pk_\Sigma}$ since
$$\mathbf{SMD}([u_i]_{pk_\Sigma}, [s'_i]_{pk_\Sigma}) = \mathbf{SMD}([0]_{pk_\Sigma}, [\alpha_i \cdot \beta_j]_{pk_\Sigma}).$$
4) In line 7, the relevance score s_i is added to I
$$[I]_{pk_\Sigma} \cdot [s_i]_{pk_\Sigma} = [I + s_i]_{pk_\Sigma}.$$

After the computation of relevance score, $E_{A_i,\gamma_j} = (\mathbb{W}_{A_i,\gamma_j}, [ID_{\gamma_j}]_{pk_{A_i}}, [K_{\gamma_j}]_{pk_{A_i}})$ is transformed into $T_{A_i,\gamma_j} = ([I_{\gamma_j}]_{pk_\Sigma}, [ID_{\gamma_j}]_{pk_{A_i}}, [K_{\gamma_j}]_{pk_{A_i}})$ (Fig. 5).

Note: The computation overhead of the **MKS** protocol does not linearly increases with $n_1 \times n_2$. The keyword equality tests executed by the **KET** protocol are independent and can be calculated in parallel. The computation time of **MKS** protocol is a little higher than that of the **KET** protocol when parallel computation is leveraged, rather than linearly grow with $n_1 \times n_2$. It is demonstrated by the performance test in Section V.

E.2. Top-k Ranking

After the relevance scores are calculated, it is necessary to find the top-k relevant encrypted files according to the scores. The requirements of the top-k ranking are listed below. (1) During the ranking process, the plaintext of the encrypted relevance score should not be revealed to both CP and CSP. (2) The CP and CSP do not know which top-k documents are returned to user.

In order to realize top-k ranking, three privacy-preserving protocols are proposed.

1) Secure maximum protocol across domains (**MAX**) will figure out the encrypted file (with the maximum relevance score) from two encrypted files.
2) Secure maximum out of n protocol across domains ($\mathbf{MAX}_n$) will utilize **MAX** protocol to find the encrypted file (with the maximum relevance score) from n encrypted files.
3) Secure top-k data retrieval protocol across domains (**Top-K**) will leverage $\mathbf{MAX}_n$ protocol to find the top-k relevant files.

E.2.1. Secure Maximum Protocol Across Domains (**MAX**)

Given $T_{A_{i_1},\gamma_{j_1}} = ([I_{\gamma_{j_1}}]_{pk_\Sigma}, [ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, [K_{\gamma_{j_1}}]_{pk_{A_{i_1}}})$ and $T_{A_{i_2},\gamma_{j_2}} = ([I_{\gamma_{j_2}}]_{pk_\Sigma}, [ID_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, [K_{\gamma_{j_2}}]_{pk_{A_{i_2}}})$ that are encrypted using different keys, **MAX** protocol will output a new tuple $T_U = ([I_U]_{pk_\Sigma}, [ID_U]_{pk_\Sigma}, [K_U]_{pk_\Sigma})$, such that $I_U = max(I_{\gamma_{j_1}}, I_{\gamma_{j_2}})$ and ID_U, K_U correspond to its file identity and file encryption key. In the protocol, CP and CSP could not distinguish the source of T_U. It has three steps and requires the interaction between CP and CSP. (The correctness of **MAX** protocol is verified in Supplemental Materials B.)

Step 1: CP computes:

$$[I'_{\gamma_{j_1}}]_{pk_\Sigma} = [I_{\gamma_{j_1}}]^2_{pk_\Sigma} \cdot [1]_{pk_\Sigma} = [2I_{\gamma_{j_1}} + 1]_{pk_\Sigma};$$
$$[I'_{\gamma_{j_2}}]_{pk_\Sigma} = [I_{\gamma_{j_2}}]^2_{pk_\Sigma} = [2I_{\gamma_{j_2}}]_{pk_\Sigma}.$$

The purpose of the calculation is that if $I_{\gamma_{j_1}} \geq I_{\gamma_{j_2}}$ and $I_{\gamma_{j_1}}, I_{\gamma_{j_2}} \geq 0$, we have $I'_{\gamma_{j_1}} > I'_{\gamma_{j_2}}$.

Then, CP randomly selects $r_1, r'_1, r_2, r_3, r_4 \in \mathbb{Z}_N$, where $\mathcal{L}(r_1) < \mathcal{L}(N)/4 - 1$, $\mathcal{L}(r'_1) < \mathcal{L}(N)/8$. Then, CP flips a random coin $s \in \{0, 1\}$.

If $s = 1$, CP and CSP jointly calculate:

$$C_1 = ([I'_{\gamma_{j_1}}]_{pk_\Sigma})^{r_1} \cdot ([I'_{\gamma_{j_2}}]_{pk_\Sigma})^{N-r_1} \cdot [r'_1]_{pk_\Sigma} \quad (1)$$
$$= [r_1(I'_{\gamma_{j_1}} - I'_{\gamma_{j_2}}) + r'_1]_{pk_\Sigma};$$
$$C_2 = [I_{\gamma_{j_2}}]_{pk_\Sigma} \cdot ([I_{\gamma_{j_1}}]_{pk_\Sigma})^{N-1} \cdot [r_2]_{pk_\Sigma} \quad (2)$$
$$= [I_{\gamma_{j_2}} - I_{\gamma_{j_1}} + r_2]_{pk_\Sigma};$$
$$C_3 = \mathbf{SAD}([ID_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, ([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}})^{N-1}) \cdot [r_3]_{pk_\Sigma} \quad (3)$$
$$= [ID_{\gamma_{j_2}} - ID_{\gamma_{j_1}} + r_3]_{pk_\Sigma};$$
$$C_4 = \mathbf{SAD}([K_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, ([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}})^{N-1}) \cdot [r_4]_{pk_\Sigma} \quad (4)$$
$$= [K_{\gamma_{j_2}} - K_{\gamma_{j_1}} + r_4]_{pk_\Sigma}.$$

If $s = 0$, CP and CSP jointly calculates:

$$C_1 = ([I'_{\gamma_{j_2}}]_{pk_\Sigma})^{r_1} \cdot ([I'_{\gamma_{j_1}}]_{pk_\Sigma})^{N-r_1} \cdot [r'_1]_{pk_\Sigma} \quad (5)$$
$$= [r_1(I'_{\gamma_{j_2}} - I'_{\gamma_{j_1}}) + r'_1]_{pk_\Sigma};$$
$$C_2 = [I_{\gamma_{j_1}}]_{pk_\Sigma} \cdot ([I_{\gamma_{j_2}}]_{pk_\Sigma})^{N-1} \cdot [r_2]_{pk_\Sigma} \quad (6)$$
$$= [I_{\gamma_{j_1}} - I_{\gamma_{j_2}} + r_2]_{pk_\Sigma};$$
$$C_3 = \mathbf{SAD}([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, ([ID_{\gamma_{j_2}}]_{pk_{A_{i_2}}})^{N-1}) \cdot [r_3]_{pk_\Sigma} \quad (7)$$
$$= [ID_{\gamma_{j_1}} - ID_{\gamma_{j_2}} + r_3]_{pk_\Sigma};$$
$$C_4 = \mathbf{SAD}([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, ([K_{\gamma_{j_2}}]_{pk_{A_{i_2}}})^{N-1}) \cdot [r_4]_{pk_\Sigma} \quad (8)$$
$$= [K_{\gamma_{j_1}} - K_{\gamma_{j_2}} + r_4]_{pk_\Sigma}.$$

CP utilizes partial strong secret key SK_1 to calculate $C'_1 = \mathbf{PD1}_{SK_1}(C_1)$ and sends C'_1, C_1, C_2, C_3, C_4 to CSP.

Step 2: After receiving C'_1, C_1, C_2, C_3, C_4, CSP calculates C''_1 by using his partial strong secret key SK_2

$$C''_1 = \mathbf{PD2}_{SK_2}(C_1, C'_1).$$

If $C''_1 < \mathcal{L}(N)/2$, CSP sets $\alpha = 0$ and computes

$$C_5 = [0]_{pk_\Sigma}, C_6 = [0]_{pk_\Sigma}, C_7 = [0]_{pk_\Sigma}.$$

If $C''_1 > \mathcal{L}(N)/2$, CSP sets $\alpha = 1$ and computes

$$C_5 = \mathbf{CR}(C_2), C_6 = \mathbf{CR}(C_3), C_7 = \mathbf{CR}(C_4).$$

Then, CSP encrypts $[\alpha]_{pk_\Sigma}$ and sends $([\alpha]_{pk_\Sigma}, C_5, C_6, C_7)$ to CP.

Step 3: When $([\alpha]_{pk_\Sigma}, C_5, C_6, C_7)$ is received, CP will execute the following computations according to the value of s tossed in step 1.

If $s = 1$, CP and CSP jointly compute

$$[I_U]_{pk_\Sigma} = [I_{\gamma_{j_1}}]_{pk_\Sigma} \cdot C_5 \cdot ([\alpha]_{pk_\Sigma})^{N-r_2};$$
$$[ID_U]_{pk_\Sigma} = \mathbf{SAD}([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, C_6) \cdot ([\alpha]_{pk_\Sigma})^{N-r_3};$$
$$[K_U]_{pk_\Sigma} = \mathbf{SAD}([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, C_7) \cdot ([\alpha]_{pk_\Sigma})^{N-r_4}.$$

If $s = 0$, CP and CSP jointly compute

$$[I_U]_{pk_\Sigma} = [I_{\gamma_{j_2}}]_{pk_\Sigma} \cdot C_5 \cdot ([\alpha]_{pk_\Sigma})^{N-r_2};$$
$$[ID_U]_{pk_\Sigma} = \mathbf{SAD}([ID_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, C_6) \cdot ([\alpha]_{pk_\Sigma})^{N-r_3};$$
$$[K_U]_{pk_\Sigma} = \mathbf{SAD}([K_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, C_7) \cdot ([\alpha]_{pk_\Sigma})^{N-r_4}.$$

It is obvious that I_U is the maximum relevance score between $I_{\gamma_{j_1}}$ and $I_{\gamma_{j_2}}$.

E.2.2. Secure Maximum Out of n Protocol Across Domains ($\mathbf{MAX}_n$)

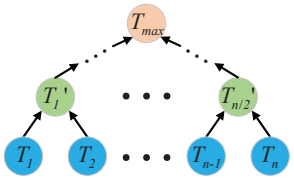


Fig. 6: Illustration of $\mathbf{MAX}_n$ Protocol

Taken as input n encrypted tuples $T_1, \cdots, T_n$, the $\mathbf{MAX}_n$ protocol could output a new tuple

$$T_{MAX} = ([I_{MAX}]_{pk_\Sigma}, [ID_{MAX}]_{pk_\Sigma}, [K_{MAX}]_{pk_\Sigma}),$$

such that $I_{MAX} = max(I_1, \cdots, I_n)$ and ID_{MAX}, K_{MAX} correspond to its file identity and file encryption key. In the protocol, CP and CSP can not recover or distinguish the source of T_{MAX}. The construction is illustrated as below.

As shown in Fig. 6, $\mathbf{MAX}_n$ protocol needs $\lceil log_2 n \rceil$ rounds to find the maximum tuple. In each round, the maximum tuple of the two adjacent encrypted tuples will be computed (use **MAX** protocol). After $\lceil log_2 n \rceil$ rounds, there is only one tuple left, which is the maximum tuple T_{MAX}.

E.2.3. Secure Top-K Data Retrieval Protocol Across Domains (**Top-K**)

Algorithm 2: SECURE TOP-K DATA RETRIEVAL PROTOCOL ACROSS DOMAINS (**Top-K**)

Input: CP has n ciphertext $T_1, \cdots, T_n, (k < n)$, where $T_i = \langle T_{i,1}, T_{i,2}, T_{i,3} \rangle = \langle [I_i]_{pk_{A_i}}, [ID_i]_{pk_{A_i}}, [K_i]_{pk_{A_i}} \rangle$.

Output: CP can get top-k file names and file secret keys (in encrypted form) corresponding to top-k relevance scores.

1 Initialize sets $S_a = \emptyset$ and $S_b = \{T_1, \cdots, T_n\}$;
2 **for** $i = 1$ *to* k **do**
3 CP and CSP jointly run $T_{MAX_i} = \mathbf{MAX}_n(T_1, \cdots, T_n)$ to get the tuple $T_{MAX_i} = \langle [I_{MAX_i}]_{pk_\Sigma}, [ID_{MAX_i}]_{pk_\Sigma}, [K_{MAX_i}]_{pk_\Sigma} \rangle$ with maximum relevance score, where $T_1, \cdots, T_n \in S_b$;
4 Insert the tuple T_{MAX_i} into set S_a;
5 **for** $j = 1$ *to* n **do**
6 CP and CSP jointly calculate: $V_j = \mathbf{SAD}(([ID_{MAX_i}]_{pk_\Sigma})^{r_j}, (T_{j,2})^{(N-r_j)})$, in which $r_j \in Z_N$ is a random number;
7 **(@CP)**: Partially decrypt V_j to $V'_j = \mathbf{PD1}_{SK_1}(V_j)$;
8 Permute (V_j, V'_j) using permutation function π_i. The result is denoted as $(V_{\pi_i(j)}, V'_{\pi_i(j)})$, which are sent to CSP;
9 **(@CSP)**: Decrypt $V''_{\pi_i(j)} = \mathbf{PD2}_{SK_2}(V_{\pi_i(j)}, V'_{\pi_i(j)})$ by using SK_2 and denote it as $\beta_j = V''_{\pi_i(j)}$;
10 **if** $\beta_j = 0$ **then**
11 denote $A_{\pi_i(j)} = [0]_{pk_\Sigma}$;
12 **else**
13 denote $A_{\pi_i(j)} = [1]_{pk_\Sigma}$;
14 Send $A_{\pi_i(j)}$ back to CP;
15 **(@CP)** CP obtains A_j by using permutation π_i^{-1};
16 CP refreshes $\{[I_1]_{pk_\Sigma}, \cdots, [I_n]_{pk_\Sigma}\} \in S_b$ by computing
17 **for** $j = 1$ *to* n **do**
18 CP and CSP jointly calculate: $[I_j]_{pk_\Sigma} = \mathbf{SMD}(T_{j,1}, A_j)$;
19 **Return** the set $S_a = (T_{MAX_1}, \cdots, T_{MAX_k})$.

On input n encrypted tuples $T_1, \cdots, T_n$, **Top-K** protocol outputs the k new encrypted tuples that have the k highest relevance scores. The protocol is illustrated as following.

Firstly, it initializes an empty set S_a to store the top-k result. The set S_b is assigned with $T_1, \cdots, T_n$. The **Top-K** protocol needs k rounds to get the result. In each round, the protocol picks up the maximum tuple. The round is illustrated in detail as following.

1) (Line 3-4) Run $\mathbf{MAX}_n$ protocol to get the maximum tuple T_{MAX_i} in the i-th round, which is inserted into S_a.
2) (Line 5-7) For each encrypted tuple in S_b, CP and CSP calculates

$$\begin{aligned} V_j &= \mathbf{SAD}(([ID_{MAX_i}]_{pk_\Sigma})^{r_j}, (T_{j,2})^{(N-r_j)}) \\ &= [r_j(ID_{MAX_i} - ID_j)]_{pk_\Sigma}. \end{aligned}$$

If $ID_j = ID_{MAX_i}$, we have $V_j = [0]_{pk_\Sigma}$. Otherwise, $V_j \neq [0]_{pk_\Sigma}$. Then, CP partially decrypts V_j and stores the result in V'_j.
3) (Line 8) To conceal the plaintext information, CP uses a permutation π_i to disturb $(V_1, \cdots, V_n)$ and $(V'_1, \cdots, V'_n)$. Then, $\{(V_{\pi_i(j)}, V'_{\pi_i(j)})\}$ for $1 \leq j \leq n$ are sends to CSP.
4) (Line 9-14) CSP decrypts $\{(V_{\pi_i(j)}, V'_{\pi_i(j)})\}$ to get β_j for $1 \leq j \leq n$. If $\beta_j = 0$, CSP sets $A_{\pi_i(j)} = [0]_{pk_\Sigma}$. Otherwise, $A_{\pi_i(j)} = [1]_{pk_\Sigma}$.
5) (Line 15) Receiving $(A_{\pi_i(1)}, \cdots, A_{\pi_i(n)})$, CP firstly utilizes π_i^{-1} recover the order and obtains $(A_1, \cdots, A_n)$. It easy to find that the origin tuple T_ζ of T_{MAX_i} will have $A_\zeta = [0]_{pk_\Sigma}$ and other tuples will have $A_j = [1]_{pk_\Sigma}$ for $1 \leq j \leq n$ and $j \neq \zeta$.
6) (Line 16-18) The $\{[I_1]_{pk_\Sigma}, \cdots, [I_n]_{pk_\Sigma}\} \in S_b$ are refreshed. $[I_\zeta]_{pk_\Sigma}$ in the origin tuple T_ζ of T_{MAX_i} will be set to $[0]_{pk_\Sigma}$ since $[I_\zeta]_{pk_\Sigma} = \mathbf{SMD}(T_{\zeta,1}, A_j) = \mathbf{SMD}([I_\zeta]_{pk_\Sigma}, [0]_{pk_\Sigma}) = [I_\zeta * 0]_{pk_\Sigma} = [0]_{pk_\Sigma}$.
For $1 \leq j \leq n$ and $j \neq \zeta$, $[I_j]_{pk_\Sigma}$ will not be changed since $[I_j]_{pk_\Sigma} = \mathbf{SMD}(T_{j,1}, A_j) = \mathbf{SMD}([I_j]_{pk_\Sigma}, [1]_{pk_\Sigma}) = [I_j * 1]_{pk_\Sigma} = [I_j]_{pk_\Sigma}$.

After k rounds computation, S_a will contain k tuples that have the k highest relevance scores.

F. Decryption

Receiving the top-k encrypted files, the user B utilizes public key pk_Σ to recover the relevance score I_i, file identity ID_i and the key K_i for $1 \leq i \leq k$. Then, using the private information retrieval (PIR) [49], [50] or oblivious RAM (ORAM) methods [51], [52], the user securely gets back the corresponding encrypted files from CP without leaking the access pattern. The key K_i could be hashed to symmetric key $K'_i = H_2(K_i) \in \mathcal{K}$, which is utilized to decrypt the file M_i.

G. Implication of the System

This system can be used in the secure storage system, such as the secure cloud storage system, encrypted healthcare record storage system, secure multimedia storage system, etc. Take the secure electronic health storage system as an example. The patients are the data owners, and the doctors and nurses are the data users. Suppose that a medical director A of the pediatric department has got the access authorization from a set of young patients (B_1, B_2, B_3). The young patient B_1 suffered from pedopneumonia with a high fever. The keywords extracted from his electronic health record (EHR) (file identity ID_1) are "pedopneumonia, fever" with keywords weights "7, 3". B_2 suffered from parascarlatina with a low fever and a slight cough, and his EHR (file identity ID_2) has keywords "parascarlatina, fever, cough" with weights "7, 1, 1". B_3 suffered from pedopneumonia with a severe cough, and his EHR (file identity ID_3) has keywords "pedopneumonia, cough" with weights "7, 5". These files and keywords and weights are encrypted using the encryption algorithm in Section IV-C.

The medical director A studies the pedopneumonia disease. He figures out a set of query keywords "pedopneumonia, fever, cough" with preference scores "5, 2, 1", and wants to get the top-2 results. The search query is encrypted using the query algorithm in Section IV-D. Then, CP executes the search algorithm in Section IV-E and returns the encrypted EHRs with file identities ID_1, ID_3. In the search process, the CP and CSP do not know which encrypted files match with the query trapdoor and which documents are returned to the user. Finally, A decrypts the result using the authorization secret key sk_Σ.

V. Performance Analysis

We conduct extensive experiments to evaluate the performance of our MRSE in this section. The experiments are performed on PC running Windows 7 64-bit operation system with the following settings: Intel(R) Core(TM) i7-4790 CPU @3.60GHz, 12GB RAM. All the protocols and algorithms are programmed using Java language and executed by Eclipse application program. We utilize multi-thread programming method to implement the system.

The length of N (denoted as $\mathcal{L}(N)$) will affect the performance of the protocols to a great extent. Both the running time and communication cost of these protocols will increase with the length of N. We utilize 512, 768, 1024, 1280, 1536, 1792 and 2048 bits to denote $\mathcal{L}(N)$ in the following experiments, respectively. When $\mathcal{L}(N) = 1024$, it achieves 80-bit security level [53]. We recommend the readers to use $\mathcal{L}(N) = 1024$ as the parameter in real applications to balance the performance and security level.

We set the file number $|F| = 128, 512, 1024, 2048, 4096$ to test the system performance. In the test, eight keywords are extracted from a file to build the encrypted index and four keyword are specified to generate a keyword trapdoor. The number of the extracted keyword and the queried keyword can be changed according to the real application. In the $Encryption$ and $Query$ algorithms, we also specify other keyword numbers to evaluate their performances.

A. Performance of Crypto Primitives and Protocols

TABLE II: **K2C** Execution Time (s)

N	512	768	1024	1280	1536	1792	2048
K2C	0.005	0.007	0.016	0.029	0.055	0.074	0.110

TABLE III: **KET** Execution Time (s)

N	512	768	1024	1280	1536	1792	2048
CP	0.049	0.162	0.333	0.671	1.036	1.637	2.450
CSP	0.005	0.019	0.039	0.078	0.126	0.198	0.297
Total	0.054	0.181	0.372	0.749	1.162	1.835	2.747

TABLE IV: **MAX** Execution Time (s)

N	512	768	1024	1280	1536	1792	2048
CP	0.059	0.201	0.360	0.723	1.319	2.026	2.729
CSP	0.007	0.023	0.042	0.084	0.160	0.244	0.330
Total	0.066	0.224	0.402	0.807	1.479	2.270	3.059

TABLE V: **MKS** Execution Time (s)

N	512	768	1024	1280	1536	1792	2048
CP	0.213	0.496	0.981	1.836	3.473	4.418	6.807
CSP	0.025	0.059	0.118	0.221	0.451	0.533	0.812
Total	0.238	0.555	1.099	2.057	3.888	4.951	7.619

Firstly, we test the performance of the basic algorithms and protocols of our proposed system on the PC in terms of computation and communication overhead. Shown in Table II-VI, all these protocols have performances increase with $\mathcal{L}(N)$. The reason is that the overhead of basic calculations (such as modular addition, multiplication and exponentiation) grow with $\mathcal{L}(N)$.

TABLE VI: Communication Overhead (KB)

N	512	768	1024	1280	1536	1792	2048
KET	4.33	6.51	8.69	10.86	13.04	15.21	17.40
MAX	1.02	1.53	2.04	2.55	3.06	3.58	4.09
MKS	78.2	117.2	156.4	195.6	234.7	274.0	313.0

TABLE VII: $\mathbf{MAX}_n$ Execution Time (s)

|F|	128	512	1024	2048	4096
CP	2.520	3.240	3.602	3.865	4.215
CSP	0.294	0.377	0.421	0.557	0.749
Total	2.814	3.617	4.023	4.422	4.964

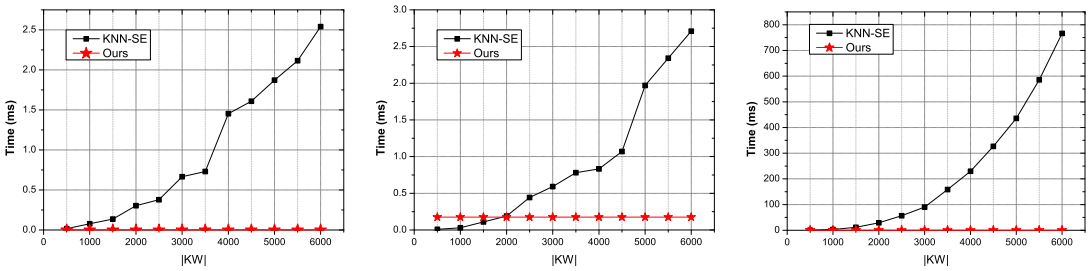
- Table II shows the running time of **K2C** algorithm. When $\mathcal{L}(N) = 1024$, **K2C** algorithm requires 0.016 s to encrypt a keyword into a ciphertext. Since this algorithm is executed by CP, there is no communication cost.
- Table III shows the computation cost of **KET** protocol. The communication cost between CP and CSP in **KET** protocol is depicted in Table VI. When $\mathcal{L}(N) = 1024$, $time_{\text{CP}} = 0.333$ s, $time_{\text{CSP}} = 0.039$ s, $time_{\text{Total}} = 0.372$ s and the communication cost is 8.69 KB. It indicates that 0.372 s running time and 8.69 KB data transmission is required for a privacy-preserving keyword equality test across domains.
- Table IV shows the running time of **MAX** protocol that varies with the bit number of N. The transmission cost is shown in Table VI. When $\mathcal{L}(N) = 1024$, $time_{\text{CP}} = 0.360$ s, $time_{\text{CSP}} = 0.042$ s, $time_{\text{Total}} = 0.402$ s and the communication cost is 2.04 KB.
- Table V and VI describe the computation and communication overheads of **MKS** protocol, in which a set of **KET** protocols are executed to test the keywords equality. These equality tests can be paralleled executed to improve the efficiency using multi-thread programming method. In the test, we set the encrypted index contain eight keywords and the search query have four keywords. When $\mathcal{L}(N) = 1024$, $time_{\text{CP}} = 0.981$ s, $time_{\text{CSP}} = 0.118$ s, $time_{\text{Total}} = 1.099$ s and the communication cost is 156.4 KB.
- Table VII indicates that the computation cost of $\mathbf{MAX}_n$ protocol increases with the number n of encrypted files. In this table, $\mathcal{L}(N) = 1024$. As shown in Figure 6, the **MAX** sub-protocol in the same level can be executed in parallel. If $|F| = n = 2^{\delta}$, the time consumed by $\mathbf{MAX}_n$ protocol approximately equals to δ times of the running

TABLE VIII: **TOP-1** Execution Time (s)

|F|	128	512	1024	2048	4096
CP	3.167	3.803	4.356	5.054	5.629
CSP	0.368	0.538	0.889	0.995	1.124
Total	3.535	4.341	5.245	6.049	6.753

TABLE IX: Communication Overhead (MB)

|F|	128	512	1024	2048	4096
$\mathbf{MAX}_n$	0.257	1.042	2.089	4.182	8.369
TOP-1	1.108	4.445	8.894	17.792	35.589



(a) **Key Generation Time** (b) **Encryption Time** (c) **Query Time**

Fig. 7: Computation Overhead Comparison

time of **MAX** sub-protocol. With the parallel computation, when $|F| = 4096$, $time_{\text{CP}} = 5.629$ s, $time_{\text{CSP}} = 1.124$ s, $time_{\text{Total}} = 6.753$ s and the communication cost is 8.369 MB (shown in Table IX).

- Table VIII shows the computation overhead of **Top-1** protocol that varies with the number $|F|$ of encrypted files. Here we set $\mathcal{L}(N) = 1024$. The most important sub-protocol in **Top-1** is the $\textbf{MAX}_n$ protocol. Simillarly, we also use parallel computations. When $|F| = n = 4096$, $time_{\text{CP}} = 5.353$ s, $time_{\text{CSP}} = 1.885$ s, $time_{\text{Total}} = 7.238$ s and the communication cost of **Top-1** is 35.589 MB (shown in Table IX). Since each loop in **Top-K** protocol is exactly the same. We only test the performance of **Top-1**. The readers can easily deduce the performance of **Top-K** by multiplying a factor k.

TABLE X: Enc or $Query$ Execution Time (s)

|KW|	**1**	**2**	**3**	**4**	**5**
Total	0.032	0.048	0.064	0.080	0.096
|KW|	**6**	**7**	**8**	**9**	**10**
Total	0.112	0.128	0.144	0.160	0.176

TABLE XI: $Search$ Execution Time (s)

K	**1**	**2**	**3**	**4**	**5**
CP	5.353	10.706	16.059	21.411	26.765
CSP	1.885	3.769	5.653	7.537	9.422
Total	7.238	14.475	21.712	28.948	36.187

TABLE XII: KNN-SE Execution Time (s)

|KW|	**KeyGen**	**BuildIndex**	**Trapdoor**	**Relevance Score Calculation**
500	0.017	0.008	0.390	0.003
1000	0.078	0.030	3.253	0.003
1500	0.136	0.110	11.398	0.003
2000	0.303	0.189	29.229	0.003
2500	0.379	0.442	56.778	0.003
3000	0.665	0.592	90.278	0.003
3500	0.730	0.781	158.527	0.003
4000	1.453	0.832	229.628	0.003
4500	1.608	1.069	326.443	0.003
5000	1.871	1.969	435.608	0.003
5500	2.115	2.341	585.812	0.003
6000	2.541	2.710	766.110	0.003

B. Performance of Multi-user MRSE

In this subsection, we test the performance of the suggested novel MRSE system. To achieve 80-bit security level, we choose the parameter $\mathcal{L}(N) = 1024$. The encrypted file number $|F| = 4096$.

- Since the encryption index generation process in $Encyption$ algorithm and query generation process in $Query$ algorithm are almost the same, the performance of these two algorithms are described in Table X. The running time of $Encyption$ and $Query$ algorithms grows with the number of keywords (denoted as $|KW|$). When $|KW| = 6$ and 10, the execution times are 0.112 s and 0.176 s, respectively.
- The $Search$ algorithm is executed by the interaction between CP and CSP. The computation time of $Search$ algorithm (shown in Table XI) varies with the number of returned results. If only one result is returned, $time_{\text{CP}} = 5.353$ s, $time_{\text{CSP}} = 1.885$ s, $time_{\text{Total}} = 7.238$ s. If five results are returned, $time_{\text{CP}} = 26.765$ s, $time_{\text{CSP}} = 9.422$ s, $time_{\text{Total}} = 36.187$ s. In order to improve user's experience, it is important to reduce the wait time of data user after he submits a search query. One optimization method is to return only one result to user each time. When the user is decrypting the received result, the following results will be continually returned to user.

The execution time of KNN-SE is shown in Table XII. KNN-SE has execution times rapidly increase with the size of predefined keywords set, which is denoted as $|KW|$. When $|KW| = 6000$, $KeyGen_{time}$ = 2.541 s, $BuildIndex_{time}$ = 2.710 s, $Trapdoor_{time}$ = 766.11 s. When $|KW|$ becomes larger, the matrixes M_1 and M_2 will increase and the time to compute M_1^{-1} and M_2^{-1} grows quickly. The data users have to consume more time to generate a search query.

The computation overhead of our system is compared with KNN-SE in Fig. 7.

- In the key generation procedure (shown in Fig. 7(a)), our system requires 4×10^{-5} s to generate the master secret key and a public/secret key pair for the user. The key generation time of KNN-SE drastically increases with the the size $|KW|$ of predefined keywords set. When $|KW| = 500$, the key generation time of KNN-SE is 0.017 s; and that is 2.541 s when $|KW| = 6000$.
- In the encryption procedure (shown in Fig. 7(b)), assume that ten keywords are extracted from the file. Our system

requires 0.176 s to build the encrypted keyword index. The build index time of KNN-SE quickly grows with $|KW|$. When $|KW| = 2000$, the build index time of KNN-SE is 0.189 s; and that is 2.710 s when $|KW| = 6000$.

- In the query procedure (shown in Fig. 7(b)), assume that ten keywords are queried. Our system requires 0.176 s to generate a trapdoor. The trapdoor generation time of KNN-SE also rapidly increases with $|KW|$. When $|KW| = 500$, the build index time of KNN-SE is 0.390 s; and that is 766.110 s when $|KW| = 6000$.
- The search time is not plotted since that of KNN-SE is constant to be 0.003 s and that of our system is also constant to be 7.238 s. Since the calculated relevance score of KNN-SE is plaintext, its time consumption is smaller than ours. Our system has better efficiency in $KeyGen, BuildIndex, Trapdoor$ algorithms.

VI. Security Analysis

In this section, we prove the security of the protocols under the attack model (defined in Section II-B) and security model (defined in Section II-C). Then, we analyze the security of the proposed novel MRSE system.

A. Protocols Security Proof

Theorem 1. *The **KET** protocol proposed in Section III-C is secure to test the equality on two keyword ciphertext in the presence of semi-honest (non-colluding) attackers* $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

Please refer Supplemental Materials C for the security proof.

Theorem 2. *The **KET** protocol is secure against the adversary* $\mathcal{A}^*$ *defined in the attack model.*

Please refer Supplemental Materials C for the security proof.

Theorem 3. *The **MKS** protocol proposed in Section IV-E is secure to calculate the relevance score on encrypted index and query in the presence of semi-honest (non-colluding) attackers* $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

Proof. **MKS** protocol calls **KET** and **SMD** as subprotocols and all the transmitted data are encrypted using PCTD encryption. Since **KET** and **SMD** protocols are proved secure in Theorem 1 and [40], the **MKS** protocol is also secure in the presence of attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$. □

Theorem 4. *The **MKS** protocol is secure against the adversary* $\mathcal{A}^*$ *defined in the attack model.*

Proof. **MKS** protocol calls **KET** and **SMD** as subprotocols and all the transmitted data are encrypted using PCTD encryption. The **KET** protocol is proved secure against the adversary $\mathcal{A}^*$ (defined in the attack model) in Theorem 2, and the **SMD** protocol is proved secure in [40]. Thus, the **MKS** protocol is also secure against the adversary $\mathcal{A}^*$ defined in the attack model. □

Theorem 5. *The **MAX** protocol proposed in Section IV-E is secure to calculate the maximum encrypted data (with highest relevance score) on two encrypted ciphertext in the presence of semi-honest (non-colluding) attackers* $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

Please refer Supplemental Materials C for the security proof.

Theorem 6. *The **MAX** protocol is secure against the adversary* $\mathcal{A}^*$ *defined in the attack model.*

Please refer Supplemental Materials C for the security proof.

Theorem 7. *The* $\mathbf{MAX}_n$ *protocol proposed in Section IV-E is secure to calculate the maximum encrypted data (with highest relevance score) on* n *encrypted ciphertext in the presence of semi-honest (non-colluding) attackers* $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

Proof. $\mathbf{MAX}_n$ protocol calls **MAX** as subprotocol and all data are encrypted using PCTD encryption. Since **MAX** protocol is proved secure in Theorem 5, the $\mathbf{MAX}_n$ protocol is also secure in the presence of attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$. □

Theorem 8. *The* $\mathbf{MAX}_n$ *protocol is secure against the adversary* $\mathcal{A}^*$ *defined in the attack model.*

Proof. $\mathbf{MAX}_n$ protocol calls **MAX** as subprotocol and all data are encrypted using PCTD encryption. Since **MAX** protocol is proved secure against the adversary $\mathcal{A}^*$ (defined in the attack model) in Theorem 6, the $\mathbf{MAX}_n$ protocol is also secure against the adversary $\mathcal{A}^*$ defined in the attack model. □

Theorem 9. *The **Top-K** protocol proposed in Section IV-E is secure to calculate the top-*k *encrypted data (with highest relevance scores) on encrypted ciphertext in the presence of semi-honest (non-colluding) attackers* $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

Please refer Supplemental Materials C for the security proof.

Theorem 10. *The **Top-K** protocol is secure against the adversary* $\mathcal{A}^*$ *defined in the attack model.*

Please refer Supplemental Materials C for the security proof.

B. System Security Analysis

In this subsection, we analyze the privacy of each algorithms in novel MRSE system.

- Key Generation: The security of the user's private key is guaranteed by discrete logarithm problem. The security of the secret keys of CP, CSP and KGC is ensured by the IND-CPA security of PCTD.
- User Authorization and Revocation: Since the users and KGC's private keys are kept secret and the signature system is cryptographically strong unforgeable, the security of authorization and revocation certificates can be guaranteed.
- Encryption: Since the keywords, file identity and symmetric key are encrypted using PCTD encryption algorithm, the encrypted index is secure. The privacy of file is ensured by the semantic security of the symmetric encryption algorithm.
- Query: The security analysis of Query algorithm is similar to Encryption algorithm.
- Search: In the search phase, the privacy-preserving relevance scores are calculated using **MKS** protocol. Then, the top-k encrypted files are calculated using **Top-K**

protocol. Since these two protocols are proved secure and all transmitted data are ciphertext, the privacy of search algorithm is ensured.

Next, we utilize the attack model in Section II-B to analyze that our system is secure against adversary $\mathcal{A}^*$.

(1) If $\mathcal{A}^*$ eavesdrops all the the contents that are transmitted between the challenge data user (including data owner) and CP. The encrypted files and indexes and the queried results can be gotten by $\mathcal{A}^*$. The intermediate calculated ciphertext that are transmitted between CP and CSP can also be eavesdropped by $\mathcal{A}^*$. Since these contents are all protected by PCTD encryption during the transmission process, $\mathcal{A}^*$ is not capable to derive the plaintext without the secret keys of data user, CP or CSP.

(2-3) Assume $\mathcal{A}^*$ compromise CP or CSP and get the partial strong private key λ_1 or λ_2. However, $\mathcal{A}^*$ can not simultaneously compromise CP and CSP. $\mathcal{A}^*$ could not get the strong secret key λ since it is randomly split into two parts using "strong secret key splitting" algorithm of PCDT. Even though $\mathcal{A}^*$ could compromise CSP and get the plaintext of intermediate result in these protocols, he is not able to get any useful information since the "blinding" method [54] is utilized in these protocols to conceal the plaintext: a random number is added to or multiplied with the message before they are sent to CSP.

(4) If the adversary $\mathcal{A}^*$ gets the secret keys of data owners or data users (except the challenge user's secret key), he could not decrypt the challenge user's ciphertext. The reason is that the private keys of different users are irrelevant.

In Theorem 1-10, the protocols **KET**, **MKS**, **MAX**, **MAX**$_n$, **Top-K** are proved secure against the adversary $\mathcal{A}^*$ defined in the attack model and the semi-honest (non-colluding) attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ defined in the security model. Thus, this system is also secure against these adversaries.

VII. Conclusion

Multi-keyword rank searchable encryption (MRSE) is a useful technique which allows data users to search over encrypted data in the cloud. Many MRSE systems have been proposed in the literature and most of them are constructed based on the KNN-SE algorithms. In this paper, we firstly pointed out several serious shortcomings of KNN-SE which limit the practical applications of the existing MRSE systems. We then proposed a new MRSE system which overcomes all the defects of the KNN-SE based MRSE systems. Our new system does not require a predefined keyword set at the system setup phase and support keyword search in arbitrary languages. The system allows flexible search authorization and time-controlled revocation. In additon, the relevance scores computed by the cloud server are in ciphertext form and the server is not able to tell which documents are the top-k results. We proved the security of the system and conducted extensive computer simulations to demonstrate its efficiency.

Acknowledgment

The authors thank the associate editor and reviewers for their constructive and generous feedback. This work is supported by National Natural Science Foundation of China under Grant No. 61402112, 61472307, 61472309, 61502086; Singapore National Research Foundation under the NCR Award Number NRF2014NCR-NCR001-012; AXA Research Fund; Open Fund Project of Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University) (No. MJUKF201734); Fujian Major Project of Regional Industry 2014H4015; and Major Science and Technology Project of Fujian Province under Grant No. 2015H6013.

References

[1] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing[J]. Communications of the ACM, 2010, 53(4): 50-58.

[2] Rittinghouse J W, Ransome J F. Cloud computing: implementation, management, and security[M]. CRC press, 2016.

[3] Hussein N H, Khalid A. A survey of Cloud Computing Security challenges and solutions[J]. International Journal of Computer Science and Information Security, 2016, 14(1): 52.

[4] Abdalla M, Bellare M, Catalano D, et al. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions[C]//Annual International Cryptology Conference. Springer Berlin Heidelberg, 2005: 205-222.

[5] Tomida K, Doi H, Mohri M, et al. Ciphertext Divided Anonymous HIBE and Its Transformation to Identity-Based Encryption with Keyword Search[J]. Journal of information processing, 2015, 23(5): 562-569.

[6] Yang Y, Ma M. Conjunctive Keyword Search With Designated Tester and Timing Enabled Proxy Re-Encryption Function for E-Health Clouds[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(4): 746-759.

[7] Wu Y, Lu X, Su J, et al. An Efficient Searchable Encryption Against Keyword Guessing Attacks for Sharable Electronic Medical Records in Cloud-based System[J]. Journal of medical systems, 2016, 40(12): 258.

[8] Yang Y. Attribute-based data retrieval with semantic keyword search for e-health cloud[J]. Journal of Cloud Computing, 2015, 4(1): 1.

[9] Wen M, Lu R, Lei J, et al. Sesa: an efficient searchable encryption scheme for auction in emerging smart grid marketing[J]. Security and Communication Networks, 2014, 7(1): 234-244.

[10] Yang Y, Zheng X, Tang C. Lightweight Distributed Secure Data Management System for Health Internet of Things[J]. Journal of Network and Computer Applications, publish online, DOI: 10.1016/j.jnca.2016.11.017.

[11] Chen R, Mu Y, Yang G, et al. Dual-server public-key encryption with keyword search for secure cloud storage[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(4): 789-798.

[12] Yang Y, Ma M. Semantic Searchable Encryption Scheme Based on Lattice in Quantum-Era[J]. Journal of Information Science and Engineering, 2016, 32(2).

[13] Golle P, Staddon J, Waters B. Secure conjunctive keyword search over encrypted data[C]//International Conference on Applied Cryptography and Network Security. Springer Berlin Heidelberg, 2004: 31-45.

[14] Byun J W, Lee D H, Lim J. Efficient conjunctive keyword search on encrypted data storage system[C]//European Public Key Infrastructure Workshop. Springer Berlin Heidelberg, 2006: 184-196.

[15] Yang Y, Ma M, Lin B. Proxy re-encryption conjunctive keyword search against keyword guessing attack[C]//Computing, Communications and IT Applications Conference (ComComAp), 2013. IEEE, 2013: 125-130.

[16] Poon H T, Miri A. An Efficient Conjunctive Keyword and Phase Search Scheme for Encrypted Cloud Storage Systems[C]//2015 IEEE 8th International Conference on Cloud Computing. IEEE, 2015: 508-515.

[17] Song D.X., Wagner D., Perrig A. Practical techniques for searches on encrypted data. IEEE Symposium on Security and Privacy 2000.

[18] Curtmola R, Garay J. A, Kamara S, Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions. In CCS 2006.

[19] Cash D, Jarecki S, Jutla C, Krawczyk H, Rosu M, Steiner M. Highly-scalable searchable symmetric encryption with support for Boolean queries. In CRYPTO 2013.

[20] Liu J. K, Au M. H, Susilo W, Liang K, Lu R, Srinivasan B. Secure sharing and searching for real-time video data in mobile cloud. IEEE Network, vol. 29, no. 2, pp. 46-50, 2015.

[21] Yang X, Lee T. T, Liu J. K, Huang X. Trust enhancement over range search for encrypted data, In IEEE TrustCom 2016, pp. 66-73, IEEE, 2016.

[22] Zuo C, Macindoe J, Yang S, Steinfeld R, Liu, J. K. Trusted boolean search on cloud using searchable symmetric encryption. IEEE TrustCom 2016, pg. 113-120. IEEE. 2016.
[23] Sun S. F, Liu J. K, Sakzad A, Steinfeld R, Yuen T H. An efficient non-interactive multi-client searchable encryption with support for boolean queries. In European Symposium on Research in Computer Security, LNCS 9878, pp. 154-172, Springer, 2016.
[24] Kermanshahi S. K, Liu J. K, Steinfeld R. Multi-user cloud-based secure keyword search. In Australasian Conference on Information Security and Privacy, LNCS 10342, pp. 227-247, Springer, 2017.
[25] Boneh D, Di Crescenzo G, Ostrovsky Rand G. Persiano G. Public key encryption with keyword search. In EUROCRYPT 2004.
[26] Liang K, Su C, Chen J, Liu J. K. Efficient multi-function data sharing and searching mechanism for cloud-based encrypted data. In ASIACCS 2016, pg. 83-94. ACM 2016.
[27] Liang K, Huang X, Guo F, Liu J. K. Privacy-preserving and regular language search over encrypted cloud data. IEEE Transactions on Information Forensics and Security, vol. 11, no. 10, pp. 2365-2376, 2016.
[28] Cao N, Wang C, Li M, et al. Privacy-preserving multi-keyword ranked search over encrypted cloud data. 2011 Proceedings IEEE INFOCOM, 2011, pp. 829 - 837, DOI: 10.1109/INFCOM.2011.5935306.
[29] Yu J, Lu P, Zhu Y, et al. Toward secure multikeyword top-k retrieval over encrypted cloud data[J]. IEEE transactions on dependable and secure computing, 2013, 10(4): 239-250.
[30] Fu Z, Sun X, Linge N, et al. Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query[J]. IEEE Transactions on Consumer Electronics, 2014, 60(1): 164-172.
[31] Fu Z, Shu J, Sun X, et al. Smart cloud search services: verifiable keyword-based semantic search over encrypted cloud data[J]. IEEE Transactions on Consumer Electronics, 2014, 60(4): 762-770.
[32] Sun W, Wang B, Cao N, et al. Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking[J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(11): 3025-3035.
[33] Li H, Liu D, Dai Y, et al. Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage[J]. IEEE Transactions on Emerging Topics in Computing, 2015, 3(1): 127-138.
[34] Li H, Liu D, Dai Y, et al. Engineering searchable encryption of mobile cloud networks: when QOE meets QOP[J]. IEEE Wireless Communications, 2015, 22(4): 74-80.
[35] Li H, Yang Y, Luan T H, et al. Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data[J]. IEEE Transactions on Dependable and Secure Computing, 2016, 13(3): 312-325.
[36] Xia Z, Wang X, Sun X, et al. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(2): 340-352.
[37] Chen C, Zhu X, Shen P, et al. An efficient privacy-preserving ranked keyword search method[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(4): 951-963.
[38] Fu Z, Wu X, Guan C, et al. Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(12): 2706-2716.
[39] Paillier P. Public-key cryptosystems based on composite degree residuosity classes[C]//International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 1999: 223-238.
[40] Liu X, Deng R, Choo K K R, et al. An Efficient Privacy-Preserving Outsourced Calculation Toolkits with Multiple Keys[J]. IEEE Transactions on Information Forensics and Security, publish online.
[41] Bresson E, Catalano D, Pointcheval D. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications[C]//International Conference on the Theory and Application of Cryptology and Information Security. Springer Berlin Heidelberg, 2003: 37-54.
[42] Liu X, Deng R H, Ding W, et al. Privacy-preserving outsourced calculation on floating point numbers[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(11): 2513-2527.
[43] Liu X, Choo R, Deng R, et al. Efficient and Privacy-Preserving Outsourced Calculation of Rational Numbers[J]. IEEE Transactions on Dependable and Secure Computing, publish online, DOI.10.1109/TDSC.2016.2536601.
[44] Do Q, Martini B, Choo K K R. A forensically sound adversary model for mobile devices[J]. PloS one, 2015, 10(9): e0138449.
[45] Kamara S, Mohassel P, Raykova M. Outsourcing Multi-Party Computation[J]. IACR Cryptology ePrint Archive, 2011, 2011: 272.
[46] Liu X, Qin B, Deng R, et al. An Efficient Privacy-Preserving Outsourced Computation over Public Data[J]. IEEE Transactions on service computing, publish online, DOI: 10.1109/TSC.2015.2511008.
[47] Wang B, Yu S, Lou W, et al. Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud[C]//IEEE INFOCOM 2014-IEEE Conference on Computer Communications. IEEE, 2014: 2112-2120.
[48] Unicode Consortium. The Unicode Standard, Version 2.0. Addison-Wesley Longman Publishing Co., Inc., 1997.
[49] Ostrovsky R, Skeith III WE. A survey of single-database private information retrieval: Techniques and applications. In PKC 2007 Apr 16 (pp. 393-411). Springer Berlin Heidelberg.
[50] Boneh D, Kushilevitz E, Ostrovsky R, Skeith III WE. Public key encryption that allows PIR queries. In CRYPTO 2007 Aug 19 (pp. 50-67). Springer Berlin Heidelberg.
[51] Stefanov E, Van Dijk M, Shi E, Fletcher C, Ren L, Yu X, Devadas S. Path ORAM: an extremely simple oblivious RAM protocol. In ACM CCS 2013 Nov 4 (pp. 299-310). ACM.
[52] Cash D, Kupcu A, Wichs D. Dynamic proofs of retrievability via oblivious ram. Journal of Cryptology. 2017 Jan 1;30(1):22-57.
[53] Barker E, Barker E, Burr W, Polk W, and Smid M, NIST special publication 800-57, NIST Special Publication, vol. 800, no. 57, pp. 1-142.
[54] Peter A, Tews E, Katzenbeisser S. Efficiently outsourcing multiparty computation under multiple keys[J]. IEEE transactions on information forensics and security, 2013, 8(12): 2046-2058.

Yang Yang received the B.Sc. degree from Xidian University, Xi'an, China, in 2006 and Ph.D. degrees from Xidian University, China, in 2012. She is a research fellow (postdoctor) under supervisor Robert H. Deng in School of Information System, Singapore Management University. She is also an associate professor in the college of mathematics and computer science, Fuzhou University. Her research interests are in the area of information security and privacy protection.

Ximeng Liu received the B.Sc. degree from Xidian University, Xi'an, China, in 2010 and Ph.D. degrees from Xidian University, China, in 2015. He was the research assistant at School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore from 2013 to 2014. Now, he is a research fellow at School of Information System, Singapore Management University, Singapore. His research interests include cloud security and big data security.

Robert H. Deng is an AXA Chair Professor of Cybersecurity in School of Information Systems, Singapore Management University. His research interests include data security and privacy, network and system security. He has served/is serving on the editorial boards of many international journals in security, such as IEEE Transactions on Information Forensics and Security, IEEE Transactions on Dependable and Secure Computing, the International Journal of Information Security, and IEEE Security and Privacy Magazine. He is Fellow of IEEE.

SUPPLEMENTAL MATERIALS

A. Overview of KNN-SE

KNN-SE is a core component in most of the existing MRSE systems. It is also referred to as secure inner product computation or secure vector space model. We will overview this method as below.

- $KeyGen(k)$. The system firstly define a keyword set $KW = (kw_1, \cdots, kw_k)$. On input the cardinality k of set KW, the algorithm will output two $k \times k$ invertible matrices (M_1, M_2) and a vector $S \in \{0,1\}^k$. The symmetric secret key of user is $sk = (M_1, M_2, S)$.
- $BuildIndex(sk, F)$. For each file F, the data owner will generate a keyword vector $I \in \{0,1\}^k$ and each bit $I(j)$ indicates whether the predefined keyword kw_j appears in F. Then, I is split into two vectors $I', I'' \in \{0,1\}^k$ following the rule: if $S(j) = 1$, $I'(j) = I''(j) = I(j)$; if $S(j) = 0$, $I'(j)$ and $I''(j)$ are set to random numbers such that $I'(j) + I''(j) = I(j)$. Then, encrypt (I', I'') with (M_1, M_2) and obtain $EI = (M_1^\top \cdot I', M_2^\top \cdot I'')$, which is the encrypted index.
- $Trapdoor(\widetilde{W})$. On input a set of keywords $\widetilde{W}$, it will generate a query vector $Q \in \{0,1\}^k$ and each bit $Q(j)$ indicates whether the predefined keyword kw_j appears in $\widetilde{W}$. Then, Q is split into two vectors $Q', Q'' \in \{0,1\}^k$ following the rule: if $S(j) = 1$, $Q'(j)$ and $Q''(j)$ are set to random numbers such that $Q'(j) + Q''(j) = Q(j)$; if $S(j) = 0$, $Q'(j) = Q''(j) = Q(j)$. Then, encrypt (I', I'') with (M_1, M_2) and obtain $EQ = (M_1^{-1} \cdot Q', M_2^{-1} \cdot Q'')$, which is the encrypted trapdoor.
- $Search(EI, EQ)$. On input encrypted index $EI = (M_1^\top \cdot I', M_2^\top \cdot I'')$ and encrypted trapdoor $EQ = (M_1^{-1} \cdot Q', M_2^{-1} \cdot Q'')$, the relevance score of file F is computed as

$$\begin{aligned} & (M_1^\top \cdot I', M_2^\top \cdot I'') \cdot (M_1^{-1} \cdot Q', M_2^{-1} \cdot Q'') \\ = \; & M_1^\top \cdot I' \cdot M_1^{-1} \cdot Q' + M_2^\top \cdot I'' \cdot M_2^{-1} \cdot Q'' \\ = \; & I'^\top \cdot Q' + I''^\top \cdot Q'' = I^\top \cdot Q. \end{aligned}$$

  After sorting these scores, the server returns the top-k files that have the highest scores.

B. Correctness of MAX

The correctness of the **MAX** protocol can be verified as below.

If $s = 1$ and $I'_{\gamma_{j_1}} > I'_{\gamma_{j_2}}$, then $C''_1 < \mathcal{L}(N)/2$, $\alpha = 0$ and $C_5 = [0]_{pk_\Sigma}, C_6 = [0]_{pk_\Sigma}, C_7 = [0]_{pk_\Sigma}$. The $T_U = ([I_U]_{pk_\Sigma}, [ID_U]_{pk_\Sigma}, [K_U]_{pk_\Sigma})$ can be computed as

$$\begin{aligned} & [I_U]_{pk_\Sigma} = [I_{\gamma_{j_1}}]_{pk_\Sigma} \cdot C_5 \cdot ([\alpha]_{pk_\Sigma})^{N-r_2} \\ = \; & [I_{\gamma_{j_1}}]_{pk_\Sigma} \cdot [0]_{pk_\Sigma} \cdot ([0]_{pk_\Sigma})^{N-r_2} = [I_{\gamma_{j_1}}]_{pk_\Sigma}; \end{aligned}$$

$$\begin{aligned} & [ID_U]_{pk_\Sigma} = \mathbf{SAD}([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, C_6) \cdot ([\alpha]_{pk_\Sigma})^{N-r_2} \\ = \; & \mathit{SAD}([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, [0]_{pk_\Sigma}) \cdot ([0]_{pk_\Sigma})^{N-r_2} \\ = \; & [ID_{\gamma_{j_1}}]_{pk_\Sigma}; \end{aligned}$$

$$\begin{aligned} & [K_U]_{pk_\Sigma} = \mathbf{SAD}([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, C_7) \cdot ([\alpha]_{pk_\Sigma})^{N-r_2} \\ = \; & \mathit{SAD}([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, [0]_{pk_\Sigma}) \cdot ([0]_{pk_\Sigma})^{N-r_2} \\ = \; & [K_{\gamma_{j_1}}]_{pk_\Sigma}. \end{aligned}$$

If $s = 1$ and $I'_A < I'_B$, then $C''_1 > \mathcal{L}(N)/2$, $\alpha = 1$ and $C_5 = \mathbf{CR}(C_2), C_6 = \mathbf{CR}(C_3), C_7 = \mathbf{CR}(C_4)$. The $T_U = ([I_U]_{pk_\Sigma}, [ID_U]_{pk_\Sigma}, [K_U]_{pk_\Sigma})$ can be computed as

$$\begin{aligned} & [I_U]_{pk_\Sigma} = [I_{\gamma_{j_1}}]_{pk_\Sigma} \cdot C_5 \cdot ([\alpha]_{pk_\Sigma})^{N-r_2} \\ = \; & [I_{\gamma_{j_1}}]_{pk_\Sigma} \cdot [I_{\gamma_{j_2}} - I_{\gamma_{j_1}} + r_2]_{pk_\Sigma} \cdot ([1]_{pk_\Sigma})^{N-r_2} \\ = \; & [I_{\gamma_{j_2}}]_{pk_\Sigma}; \end{aligned}$$

$$\begin{aligned} & [ID_U]_{pk_\Sigma} = \mathbf{SAD}([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, C_6) \cdot ([\alpha]_{pk_\Sigma})^{N-r_3} \\ = \; & \mathbf{SAD}([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, [ID_{\gamma_{j_2}} - ID_{\gamma_{j_1}} + r_3]_{pk_\Sigma}) \cdot ([1]_{pk_\Sigma})^{N-r_3} \\ = \; & [ID_{\gamma_{j_2}}]_{pk_\Sigma}; \end{aligned}$$

$$\begin{aligned} & [K_U]_{pk_\Sigma} = \mathbf{SAD}([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, C_7) \cdot ([\alpha]_{pk_\Sigma})^{N-r_4} \\ = \; & \mathbf{SAD}([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, [K_{\gamma_{j_2}} - K_{\gamma_{j_1}} + r_4]_{pk_\Sigma}) \cdot ([1]_{pk_\Sigma})^{N-r_4} \\ = \; & [K_{\gamma_{j_2}}]_{pk_\Sigma}. \end{aligned}$$

When $s = 0$, $T_U = ([I_U]_{pk_\Sigma}, [ID_U]_{pk_\Sigma}, [K_U]_{pk_\Sigma})$ can be similarly computed. The correctness verification demonstrates that the **MAX** protocol could output a new tuple T_U (with the maximum relevance score) from the two encrypted tuples.

C. Security Proof

Theorem 1. *The **KET** protocol proposed in Section III-C is secure to test the equality on two keyword ciphertext in the presence of semi-honest (non-colluding) attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.*

Proof. We now construct the following independent simulators $(Sim_{D_1}, Sim_{S_1}, Sim_{S_2})$.

Sim_{D_1} receives X and Y as input and simulates $\mathcal{A}_{D_1}$ as following. It generates the ciphertext $[X]_{pk_A}$ and $[Y]_{pk_B}$ of X and Y, respectively. The entire view of $\mathcal{A}_{D_1}$ is the encrypted data, which in both real and ideal executions are indistinguishable due to the IND-CPA security of PCTD [41].

Sim_{S_1} simulates $\mathcal{A}_{S_1}$ as following. It randomly selects $\hat{X}, \hat{Y} \in Z_N$ and encrypts them into $[\hat{X}]_{pk_A}$ and $[\hat{Y}]_{pk_B}$. Then, it computes $[\hat{X}_1]_{pk_A} = ([\hat{X}]_{pk_A})^2 \cdot [1]_{pk_A}$, $[\hat{Y}_1]_{pk_B} = ([\hat{Y}]_{pk_B})^2$, $[\hat{X}_2]_{pk_A} = ([\hat{X}]_{pk_A})^2$, $[\hat{Y}_2]_{pk_B} = ([\hat{Y}]_{pk_B})^2 \cdot [1]_{pk_B}$. According to the randomly flipped coins $s_1, s_2 \in \{0,1\}$, it inputs them into $Sim_{S_1}^{\mathbf{SAD}}$ in Ref. [40] and gets l_1, l_2. Using **PD1** algorithm, it computes l'_1, l'_2. It also randomly selects $\hat{u}'_1, \hat{u}'_2 \in \{0,1\}$ and computes $[\hat{u}'_1]_{pk_\Sigma}, [\hat{u}'_2]_{pk_\Sigma}$. According to the randomly flipped coins $s_1, s_2 \in \{0,1\}$, it calculates $[\hat{u}_1]_{pk_\Sigma}, [\hat{u}_2]_{pk_\Sigma}$ and inputs them into $Sim_{S_1}^{\mathbf{SMD}}$ in Ref. [40] and gets $[\hat{u}^*]_{pk_\Sigma}$. Then, Sim_{S_1} sends (l_1, l_2, l'_1, l'_2) and the intermediate encrypted data of $Sim_{S_1}^{\mathbf{SAD}}$ and $Sim_{S_1}^{\mathbf{SMD}}$ to $\mathcal{A}_{S_1}$. If $\mathcal{A}_{S_1}$ replies with $\perp$, then Sim_{S_1} outputs $\perp$. The IND-CPA security of PCTD ensures that $\mathcal{A}_{S_1}$'s view is indistinguishable from its view in the real world execution.

Sim_{S_2} simulates $\mathcal{A}_{S_2}$ as following. It selects random $\hat{u}'_1, \hat{u}'_2 \in \{0,1\}$. It encrypts them into $[\hat{u}'_1]_{pk_\Sigma}, [\hat{u}'_2]_{pk_\Sigma}$, which are sent to $\mathcal{A}_{S_2}$. If $\mathcal{A}_{S_2}$ replies with $\perp$, then Sim_{S_2} outputs $\perp$. This is ensured in real world due to the IND-CPA security of PCTD. In both real and ideal world, the views of $\mathcal{A}_{S_2}$ are indistinguishable. □

Theorem 2. *The **KET** protocol is secure against the adversary $\mathcal{A}^*$ defined in the attack model.*

Proof. The adversary $\mathcal{A}^*$ is assumed to have the following abilities.

(1) $\mathcal{A}^*$ is assumed to be an outside adversary and eavesdrop all the communications to get the transmitted information. As $\mathcal{A}^*$ is assumed to be an outside adversary, $\mathcal{A}^*$ cannot get data owner A's private key sk_A, data user B's private key sk_B and B's authorization secret key sk_Σ. $\mathcal{A}^*$ also cannot get CP's partial strong key SK_1 and CSP's partial strong key SK_2.

If $\mathcal{A}^*$ eavesdrops the communication channel between the system user and CP, $\mathcal{A}^*$ could get the ciphertexts $[X]_{pk_A}$ and $[Y]_{pk_B}$ that are transmitted at the beginning of the **KET** protocol, and the encrypted result $[u^*]_{pk_\Sigma}$ that is transmitted at the end of the protocol. Since $[X]_{pk_A}$, $[Y]_{pk_B}$ and $[u^*]_{pk_\Sigma}$ are encrypted using the PCTD algorithm, the adversary $\mathcal{A}^*$ cannot recover X, Y, u^* due to the IND-CPA security of PCTD.

If $\mathcal{A}^*$ eavesdrops the communication channel between CP and CSP, $\mathcal{A}^*$ could get (l_1, l'_1, l_2, l'_2) in the end of step 1, and $([u'_1]_{pk_\Sigma}, [u'_2]_{pk_\Sigma})$ in the end of step 2. In the **KET** protocol,

$$\begin{aligned} l_1 &= [\gamma_1]_{pk_\Sigma} \cdot [r_3]_{pk_\Sigma} = [\gamma_1 + r_3]_{pk_\Sigma}, \\ l_2 &= [\gamma_2]_{pk_\Sigma} \cdot [r_4]_{pk_\Sigma} = [\gamma_2 + r_4]_{pk_\Sigma}, \end{aligned}$$

$l'_1 = \mathbf{PD1}_{SK_1}(l_1)$ and $l'_2 = \mathbf{PD1}_{SK_1}(l_2)$. Since the adversary $\mathcal{A}^*$ does know the data owner B's authorization secret key sk_Σ and the CSP's partial strong key SK_2, $\mathcal{A}^*$ cannot recover the plaintexts $\gamma_1 + r_3$, $\gamma_2 + r_4$. Then, $\mathcal{A}^*$ cannot deduce the plaintexts X, Y and their relationship.

(2) $\mathcal{A}^*$ is assumed to compromise CP and get CP's partial strong key SK_1. But $\mathcal{A}^*$ cannot get CSP's partial strong key SK_2. $\mathcal{A}^*$ also cannot get data owner A's private key sk_A, data user B's private key sk_B and B's authorization secret key sk_Σ.
In step 1 of the **KET** protocol, $\mathcal{A}^*$ obtains $[X]_{pk_A}$ and $[Y]_{pk_B}$ from the data owner A and data user B. $\mathcal{A}^*$ cannot recover X nor Y without the secret keys sk_A, sk_B. In step 3, $\mathcal{A}^*$ obtains $([u'_1]_{pk_\Sigma}, [u'_2]_{pk_\Sigma})$ from CSP. Since sk_Σ is unknown, $\mathcal{A}^*$ cannot derive (u'_1, u'_2).

(3) $\mathcal{A}^*$ is assumed to compromise CSP and get CSP's partial strong key SK_2. But $\mathcal{A}^*$ cannot get CP's partial strong key SK_1. $\mathcal{A}^*$ also cannot get data owner A's private key sk_A, data user B's private key sk_B and B's authorization secret key sk_Σ.
In step 2 of the **KET** protocol, $\mathcal{A}^*$ obtains (l_1, l'_1, l_2, l'_2) transmitted by CP. Since CSP's partial strong key SK_2 is known, $\mathcal{A}^*$ decrypts

$$\begin{aligned} l''_1 &= \mathbf{PD2}_{SK_2}(l_1, l'_1) = \gamma_1 + r_3, \\ l''_2 &= \mathbf{PD2}_{SK_2}(l_2, l'_2) = \gamma_2 + r_4. \end{aligned}$$

If $\mathcal{L}(l''_1) > \mathcal{L}(N)/2$, CSP sets $u'_1 = 0$ and $u'_1 = 1$ otherwise. If $\mathcal{L}(l''_2) > \mathcal{L}(N)/2$, CSP sets $u'_2 = 0$ and $u'_2 = 1$ otherwise. Although $\mathcal{A}^*$ can get $\gamma_1 + r_3$, $\gamma_2 + r_4$, u'_1 and u'_2, $\mathcal{A}^*$ cannot deduce the relationship of size between X and Y. The reason is explained below.
In step 1, the CP flips random coins $s_1, s_2 \in \{0,1\}$ and calculates $([\gamma_1]_{pk_\Sigma}, [\gamma_2]_{pk_\Sigma})$ according to (s_1, s_2).
If $s_1 = 1$,

$$\begin{aligned} [\gamma_1]_{pk_\Sigma} &= \mathbf{SAD}(([X_1]_{pk_A})^{r_1}, ([Y_1]_{pk_B})^{N-r_1}) \\ &= [r_1(X_1 - Y_1)]_{pk_\Sigma}. \end{aligned}$$

If $s_1 = 0$,

$$\begin{aligned} [\gamma_1]_{pk_\Sigma} &= \mathbf{SAD}(([X_1]_{pk_A})^{N-r_1}, ([Y_1]_{pk_B})^{r_1}) \\ &= [r_1(Y_1 - X_1)]_{pk_\Sigma}. \end{aligned}$$

If $s_2 = 1$,

$$\begin{aligned} [\gamma_2]_{pk_\Sigma} &= \mathbf{SAD}(([X_2]_{pk_A})^{N-r_2}, ([Y_2]_{pk_B})^{r_2}) \\ &= [r_2(Y_2 - X_2)]_{pk_\Sigma}. \end{aligned}$$

If $s_2 = 0$,

$$\begin{aligned} [\gamma_2]_{pk_\Sigma} &= \mathbf{SAD}(([X_2]_{pk_A})^{r_2}, ([Y_2]_{pk_B})^{N-r_2}) \\ &= [r_2(X_2 - Y_2)]_{pk_\Sigma}. \end{aligned}$$

Then, the adversary $\mathcal{A}^*$ gets

$$l''_1 = \gamma_1 + r_3 = \begin{cases} r_1(X_1 - Y_1) + r_3, & if\ s_1 = 1, \\ r_1(Y_1 - X_1) + r_3, & if\ s_1 = 0, \end{cases}$$

$$l''_2 = \gamma_2 + r_4 = \begin{cases} r_2(Y_2 - X_2) + r_4, & if\ s_2 = 1, \\ r_2(X_2 - Y_2) + r_4, & if\ s_2 = 0. \end{cases}$$

Due to the randomness of s_1, s_2, the adversary $\mathcal{A}^*$ cannot deduce the relationship of size between X_1 and Y_1, and that between X_2 and Y_2. Thus, $\mathcal{A}^*$ cannot deduce the relationship of size between X and Y.

(4) $\mathcal{A}^*$ is assumed to be a set of collude malicious users $(B_1, \cdots, B_n)$ (except the challenge user B^*), and $\mathcal{A}^*$ gets their secret keys $(sk_{B_1}, \cdots, sk_{B_n})$. $\mathcal{A}^*$ wants to get the information that belongs to the challenge user B^*. Suppose the compared ciphertexts are $([X]_{pk_A}, [Y]_{pk_{B^*}})$, and the returned result is $[u^*]_{pk_{\Sigma^*}}$, where pk_{Σ^*} is the authorize public key from data owner A to challenge user B^*. Since the user's secret keys are independently generated, the adversary $\mathcal{A}^*$ cannot utilize $(sk_{B_1}, \cdots, sk_{B_n})$ to deduce the challenge user B^*'s secret key sk_{B^*}. $\mathcal{A}^*$ also cannot get the authorization secret key sk_{Σ^*}. Thus, $\mathcal{A}^*$ cannot recover Y nor u^*.

According to the above analysis, the **KET** protocol is secure against the adversary $\mathcal{A}^*$ defined in the attack model. □

Theorem 5. *The **MAX** protocol proposed in Section IV-E is secure to calculate the maximum encrypted data (with highest*

relevance score) on two encrypted ciphertext in the presence of semi-honest (non-colluding) attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

Proof. We now construct the following independent simulators $(Sim_{D_1}, Sim_{S_1}, Sim_{S_2})$.

Sim_{D_1} receives tuples (I_1, ID_1, K_1) and (I_2, ID_2, K_2) as input and simulates $\mathcal{A}_{D_1}$ as following. It generates the ciphertext $([I_1]_{pk_\Sigma}, [ID_1]_{pk_A}, [K_1]_{pk_A})$ and $([I_2]_{pk_\Sigma}, [ID_2]_{pk_B}, [K_2]_{pk_B})$. The entire view of $\mathcal{A}_{D_1}$ is the received tuples and the encrypted data. The IND-CPA security of PCTD ensures that $\mathcal{A}_{D_1}$'s view is indistinguishable from its view in the real world execution.

Sim_{S_1} simulates $\mathcal{A}_{S_1}$ as following. It randomly selects $(\hat{I}_1, \hat{ID}_1, \hat{K}_1)$ and $(\hat{I}_2, \hat{ID}_2, \hat{K}_2)$ and encrypts them into $([\hat{I}_1]_{pk_\Sigma}, [\hat{ID}_1]_{pk_A}, [\hat{K}_1]_{pk_A})$ and $([\hat{I}_2]_{pk_\Sigma}, [\hat{ID}_2]_{pk_B}, [\hat{K}_2]_{pk_B})$. Then, it computes $[\hat{I}_1]_{pk_\Sigma} = ([\hat{I}_1]_{pk_\Sigma})^2 \cdot [1]_{pk_\Sigma}$, $[\hat{I}_2]_{pk_\Sigma} = ([\hat{I}_2]_{pk_\Sigma})^2$. It flips a random coin $s \in \{0,1\}$. If $s = 1$, Sim_{S_1} computes $\hat{C}_1, \hat{C}_2$ using (1)-(2). Otherwise, it computes $\hat{C}_1, \hat{C}_2$ using (5)-(6). According to the randomly flipped coins $s \in \{0,1\}$, it inputs $(\hat{ID}_1, \hat{ID}_2)$ and $(\hat{K}_1, \hat{K}_2)$ into $Sim_{S_1}^{\texttt{SAD}}$ in Ref. [40] and gets $\hat{C}_3, \hat{C}_4$. Then, utilizing **PD1** algorithm, it computes $\hat{C}_1'$. It randomly selects $\hat{\alpha} \in \{0,1\}$ and computes $[\hat{\alpha}]_{pk_\Sigma}$. It generates random ciphertext $\hat{C}_6, \hat{C}_7$. According to the randomly flipped coins $s \in \{0,1\}$, it inputs them into $Sim_{S_1}^{\texttt{SAD}}$ in Ref. [40] and gets $[\hat{ID}_U]_{pk_\Sigma}, [\hat{K}_U]_{pk_\Sigma}$. Then, Sim_{S_1} sends $(\hat{C}_1, \hat{C}_2, \hat{C}_3, \hat{C}_4, \hat{C}_1')$ and the intermediate encrypted data of $Sim_{S_1}^{\texttt{SAD}}$ to $\mathcal{A}_{S_1}$. If $\mathcal{A}_{S_1}$ replies with $\perp$, then Sim_{S_1} outputs $\perp$. The IND-CPA security of PCTD ensures that $\mathcal{A}_{D_1}$'s view is indistinguishable from its view in the real world execution.

Sim_{S_2} simulates $\mathcal{A}_{S_2}$ as following. It selects random $\hat{\alpha} \in \{0,1\}$. If $\hat{\alpha} = 0$, it calculates $\hat{C}_5 = [0]_{pk_\Sigma}$, $\hat{C}_6 = [0]_{pk_\Sigma}$, $\hat{C}_7 = [0]_{pk_\Sigma}$. If $\hat{\alpha} = 1$, it generates random encryptions to be $(\hat{C}_5, \hat{C}_6, \hat{C}_7)$. For a certain $\hat{\alpha}$, the generated ciphertexts are computationally indistinguishable from the real world due to the IND-CPA security of PCTD. In both real and ideal world, the views of $\mathcal{A}_{S_2}$ are indistinguishable. □

Theorem 6. *The **MAX** protocol is secure against the adversary $\mathcal{A}^*$ defined in the attack model.*

Proof. The adversary $\mathcal{A}^*$ is assumed to have the following abilities.

(1) $\mathcal{A}^*$ is assumed to be an outside adversary and eavesdrop all the communications to get the transmitted information. As $\mathcal{A}^*$ is assumed to be an outside adversary, $\mathcal{A}^*$ cannot get data owner A's private key sk_A, data user B's private key sk_B and B's authorization secret key sk_Σ. $\mathcal{A}^*$ also cannot get CP's partial strong key SK_1 and CSP's partial strong key SK_2.
If $\mathcal{A}^*$ eavesdrops the communication channel between the system user and CP, $\mathcal{A}^*$ could get the ciphertexts $T_{A_{i_1},\gamma_{j_1}} = ([I_{\gamma_{j_1}}]_{pk_\Sigma}, [ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, [K_{\gamma_{j_1}}]_{pk_{A_{i_1}}})$ and $T_{A_{i_2},\gamma_{j_2}} = ([I_{\gamma_{j_2}}]_{pk_\Sigma}, [ID_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, [K_{\gamma_{j_2}}]_{pk_{A_{i_2}}})$ that are transmitted at the beginning of the **MAX** protocol, and the encrypted result $T_U = ([I_U]_{pk_\Sigma}, [ID_U]_{pk_\Sigma}, [K_U]_{pk_\Sigma})$ that is transmitted at the end of the protocol. Since $T_{A_{i_1},\gamma_{j_1}}$, $T_{A_{i_2},\gamma_{j_2}}$ and T_U are encrypted using the PCTD algorithm, the adversary $\mathcal{A}^*$ cannot recover $(I_{\gamma_{j_1}}, ID_{\gamma_{j_1}}, K_{\gamma_{j_1}})$, $(I_{\gamma_{j_2}}, ID_{\gamma_{j_2}}, K_{\gamma_{j_2}})$ and (I_U, ID_U, K_U) due to the IND-CPA security of PCTD.
If $\mathcal{A}^*$ eavesdrops the communication channel between CP and CSP, $\mathcal{A}^*$ could get $(C_1', C_1, C_2, C_3, C_4)$ in the end of step 1, and $([\alpha]_{pk_\Sigma}, C_5, C_6, C_7)$ in the end of step 2. In the **MAX** protocol, these ciphertexts $(C_1', C_1, C_2, C_3, C_4, [\alpha]_{pk_\Sigma}, C_5, C_6, C_7)$ are all encrypted using the public key pk_Σ. Since the adversary $\mathcal{A}^*$ does know the data owner B's authorization secret key sk_Σ and the CSP's partial strong key SK_2, $\mathcal{A}^*$ cannot recover the underlying plaintexts.

(2) $\mathcal{A}^*$ is assumed to compromise CP and get CP's partial strong key SK_1. But $\mathcal{A}^*$ cannot get CSP's partial strong key SK_2. $\mathcal{A}^*$ also cannot get data owner A's private key sk_A, data user B's private key sk_B and B's authorization secret key sk_Σ.
In step 1 of the **MAX** protocol, $\mathcal{A}^*$ obtains $T_{A_{i_1},\gamma_{j_1}}$ and $T_{A_{i_2},\gamma_{j_2}}$. $\mathcal{A}^*$ cannot recover $(I_{\gamma_{j_1}}, ID_{\gamma_{j_1}}, K_{\gamma_{j_1}})$, $(I_{\gamma_{j_2}}, ID_{\gamma_{j_2}}, K_{\gamma_{j_2}})$ without the secret keys $sk_\Sigma, sk_{A_{i_1}}, sk_{A_{i_2}}$. In step 3, $\mathcal{A}^*$ obtains $([\alpha]_{pk_\Sigma}, C_5, C_6, C_7)$ from CSP.
If $s = 1$,

$$
\begin{aligned}
& [I_U]_{pk_\Sigma} \\
= \; & [I_{\gamma_{j_1}}]_{pk_\Sigma} \cdot C_5 \cdot ([\alpha]_{pk_\Sigma})^{N-r_2} \\
= \; & \begin{cases} [I_{\gamma_{j_1}} + (I_{\gamma_{j_2}} - I_{\gamma_{j_1}} + r_2) - r_2]_{pk_\Sigma}, & if\ \alpha = 1, \\ [I_{\gamma_{j_1}} + 0 - 0]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases} \\
= \; & \begin{cases} [I_{\gamma_{j_2}}]_{pk_\Sigma}, & if\ \alpha = 1, \\ [I_{\gamma_{j_1}}]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases} \\
& [ID_U]_{pk_\Sigma} \\
= \; & \texttt{SAD}([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, C_6) \cdot ([\alpha]_{pk_\Sigma})^{N-r_3} \\
= \; & \begin{cases} [ID_{\gamma_{j_1}} + (ID_{\gamma_{j_2}} - ID_{\gamma_{j_1}} + r_3) - r_3]_{pk_\Sigma}, & if\ \alpha = 1, \\ [ID_{\gamma_{j_1}} + 0 - 0]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases} \\
= \; & \begin{cases} [ID_{\gamma_{j_2}}]_{pk_\Sigma}, & if\ \alpha = 1, \\ [ID_{\gamma_{j_1}}]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases} \\
& [K_U]_{pk_\Sigma} \\
= \; & \texttt{SAD}([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, C_7) \cdot ([\alpha]_{pk_\Sigma})^{N-r_4} \\
= \; & \begin{cases} [K_{\gamma_{j_1}} + (K_{\gamma_{j_2}} - K_{\gamma_{j_1}} + r_4) - r_4]_{pk_\Sigma}, & if\ \alpha = 1, \\ [K_{\gamma_{j_1}} + 0 - 0]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases} \\
= \; & \begin{cases} [K_{\gamma_{j_2}}]_{pk_\Sigma}, & if\ \alpha = 1, \\ [K_{\gamma_{j_1}}]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases}
\end{aligned}
$$

If $s = 0$,

$$
\begin{aligned}
& [I_U]_{pk_\Sigma} \\
= \; & [I_{\gamma_{j_2}}]_{pk_\Sigma} \cdot C_5 \cdot ([\alpha]_{pk_\Sigma})^{N-r_2} \\
= \; & \begin{cases} [I_{\gamma_{j_2}} + (I_{\gamma_{j_1}} - I_{\gamma_{j_2}} + r_2) - r_2]_{pk_\Sigma}, & if\ \alpha = 1, \\ [I_{\gamma_{j_2}} + 0 - 0]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases} \\
= \; & \begin{cases} [I_{\gamma_{j_1}}]_{pk_\Sigma}, & if\ \alpha = 1, \\ [I_{\gamma_{j_2}}]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
& [ID_U]_{pk_\Sigma} \\
= \ & \mathbf{SAD}([ID_{\gamma_{j_2}}]_{pk_{A_{i_1}}}, C_6) \cdot ([\alpha]_{pk_\Sigma})^{N-r_3} \\
= \ & \begin{cases} [ID_{\gamma_{j_2}} + (ID_{\gamma_{j_1}} - ID_{\gamma_{j_1}} + r_3) - r_3]_{pk_\Sigma}, & if\ \alpha = 1, \\ [ID_{\gamma_{j_2}} + 0 - 0]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases} \\
= \ & \begin{cases} [ID_{\gamma_{j_1}}]_{pk_\Sigma}, & if\ \alpha = 1, \\ [ID_{\gamma_{j_2}}]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
& [K_U]_{pk_\Sigma} \\
= \ & \mathbf{SAD}([K_{\gamma_{j_2}}]_{pk_{A_{i_1}}}, C_7) \cdot ([\alpha]_{pk_\Sigma})^{N-r_4} \\
= \ & \begin{cases} [K_{\gamma_{j_2}} + (K_{\gamma_{j_1}} - K_{\gamma_{j_2}} + r_4) - r_4]_{pk_\Sigma}, & if\ \alpha = 1, \\ [K_{\gamma_{j_2}} + 0 - 0]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases} \\
= \ & \begin{cases} [K_{\gamma_{j_1}}]_{pk_\Sigma}, & if\ \alpha = 1, \\ [K_{\gamma_{j_2}}]_{pk_\Sigma}, & if\ \alpha = 0, \end{cases}
\end{aligned} \tag{4}
$$

Since α is unknown to CP, $\mathcal{A}^*$ cannot decide the result T_U comes from $T_{A_{i_1},\gamma_{j_1}}$ or $T_{A_{i_2},\gamma_{j_2}}$.

(3) $\mathcal{A}^*$ is assumed to compromise CSP and get CSP's partial strong key SK_2. But $\mathcal{A}^*$ cannot get CP's partial strong key SK_1. $\mathcal{A}^*$ also cannot get data owner A's private key sk_A, data user B's private key sk_B and B's authorization secret key sk_Σ.
In step 2 of the **MAX** protocol, $\mathcal{A}^*$ obtains $(C'_1, C_1, C_2, C_3, C_4)$ transmitted by CP. Since $\mathcal{A}^*$ knows CSP's partial strong key SK_2, $\mathcal{A}^*$ decrypts $C''_1 = \mathbf{PD2}_{SK_2}(C_1, C'_1)$. If $C''_1 < \mathcal{L}(N)/2$, CSP sets $\alpha = 0$ and computes

$$C_5 = [0]_{pk_\Sigma}, C_6 = [0]_{pk_\Sigma}, C_7 = [0]_{pk_\Sigma}.$$

If $C''_1 > \mathcal{L}(N)/2$, CSP sets $\alpha = 1$ and computes

$$C_5 = \mathbf{CR}(C_2), C_6 = \mathbf{CR}(C_3), C_7 = \mathbf{CR}(C_4).$$

Although $\mathcal{A}^*$ can get the plaintext C''_1, $\mathcal{A}^*$ cannot deduce the relationship of size between $I_{\gamma_{j_1}}$ and $I_{\gamma_{j_2}}$. The reason is explained below.
In step 1, the CP flips a random coins $s \in \{0, 1\}$ and calculates (C_1, C_2, C_3, C_4) according to s.
If $s = 1$,

$$
\begin{aligned}
C_1 &= [r_1(I'_{\gamma_{j_1}} - I'_{\gamma_{j_2}}) + r'_1]_{pk_\Sigma}; \\
C_2 &= [I_{\gamma_{j_2}} - I_{\gamma_{j_1}} + r_2]_{pk_\Sigma}; \\
C_3 &= [ID_{\gamma_{j_2}} - ID_{\gamma_{j_1}} + r_3]_{pk_\Sigma}; \\
C_4 &= [K_{\gamma_{j_2}} - K_{\gamma_{j_1}} + r_4]_{pk_\Sigma}.
\end{aligned}
$$

If $s = 0$,

$$
\begin{aligned}
C_1 &= [r_1(I'_{\gamma_{j_2}} - I'_{\gamma_{j_1}}) + r'_1]_{pk_\Sigma}; \\
C_2 &= [I_{\gamma_{j_1}} - I_{\gamma_{j_2}} + r_2]_{pk_\Sigma}; \\
C_3 &= [ID_{\gamma_{j_1}} - ID_{\gamma_{j_2}} + r_3]_{pk_\Sigma}; \\
C_4 &= [K_{\gamma_{j_1}} - K_{\gamma_{j_2}} + r_4]_{pk_\Sigma}.
\end{aligned}
$$

Then, the adversary $\mathcal{A}^*$ gets

$$
\begin{aligned}
C_1 &= \begin{cases} [r_1(I'_{\gamma_{j_1}} - I'_{\gamma_{j_2}}) + r'_1]_{pk_\Sigma}, & if\ s = 1, \\ [r_1(I'_{\gamma_{j_2}} - I'_{\gamma_{j_1}}) + r'_1]_{pk_\Sigma}, & if\ s = 0, \end{cases} \\
C_2 &= \begin{cases} [I_{\gamma_{j_2}} - I_{\gamma_{j_1}} + r_2]_{pk_\Sigma}, & if\ s = 1, \\ [I_{\gamma_{j_1}} - I_{\gamma_{j_2}} + r_2]_{pk_\Sigma}, & if\ s = 0, \end{cases} \\
C_3 &= \begin{cases} [ID_{\gamma_{j_2}} - ID_{\gamma_{j_1}} + r_3]_{pk_\Sigma}, & if\ s = 1, \\ [ID_{\gamma_{j_1}} - ID_{\gamma_{j_2}} + r_3]_{pk_\Sigma}, & if\ s = 0, \end{cases} \\
C_4 &= \begin{cases} [K_{\gamma_{j_2}} - K_{\gamma_{j_1}} + r_4]_{pk_\Sigma}, & if\ s = 1, \\ [K_{\gamma_{j_1}} - K_{\gamma_{j_2}} + r_4]_{pk_\Sigma}, & if\ s = 0, \end{cases}
\end{aligned}
$$

Due to the randomness of s, the adversary $\mathcal{A}^*$ cannot deduce the relationship of size between $I_{\gamma_{j_1}}$ and $I_{\gamma_{j_2}}$.
$\mathcal{A}^*$ is assumed to be a set of collude malicious users $(B_1, \cdots, B_n)$ (except the challenge user B^*), and $\mathcal{A}^*$ gets their secret keys $(sk_{B_1}, \cdots, sk_{B_n})$. $\mathcal{A}^*$ wants to get the information that belongs to the challenge user B^*. Suppose the returned result is $T_U = ([I_U]_{pk_{\Sigma^*}}, [ID_U]_{pk_{\Sigma^*}}, [K_U]_{pk_{\Sigma^*}})$, where pk_{Σ^*} is the authorize public key from data owner A to challenge user B^*. Since the user's secret keys are independently generated, the adversary $\mathcal{A}^*$ cannot utilize $(sk_{B_1}, \cdots, sk_{B_n})$ to deduce the challenge user B^*'s secret key sk_{B^*}. $\mathcal{A}^*$ also cannot get the authorization secret key sk_{Σ^*}. Thus, $\mathcal{A}^*$ cannot recover (I_U, ID_U, K_U).

According to the above analysis, the **MAX** protocol is secure against the adversary $\mathcal{A}^*$ defined in the attack model. □

Theorem 9. *The **Top-K** protocol proposed in Section IV-E is secure to calculate the top-k encrypted data (with highest relevance scores) on encrypted ciphertext in the presence of semi-honest (non-colluding) attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.*

Proof. We now construct the following independent simulators $(Sim_{D_1}, Sim_{S_1}, Sim_{S_2})$.

Sim_{D_1} receives tuples (I_i, ID_i, K_i) for $1 \le i \le n$ as input and simulates $\mathcal{A}_{D_1}$ as following. It generates the ciphertext $(T_1, \cdots, T_n)$, where $T_i = ([I_i]_{pk_\Sigma}, [ID_i]_{pk_A}, [K_i]_{pk_A})$ for $1 \le i \le n$. The entire view of $\mathcal{A}_{D_1}$ is the received tuples and the encrypted data. The IND-CPA security of PCTD ensures that $\mathcal{A}_{D_1}$'s view is indistinguishable from its view in the real world execution.

Sim_{S_1} simulates $\mathcal{A}_{S_1}$ as following. It randomly generates $(T_1, \cdots, T_n)$ and inputs them into $Sim_{S_1}^{\mathbf{MAX}_n}$ in Theorem 7 and gets T_{MAX_i}, where $T_{MAX_i} = \langle [I_{MAX_i}]_{pk_\Sigma}, [ID_{MAX_i}]_{pk_\Sigma}, [K_{MAX_i}]_{pk_\Sigma} \rangle$. It inserts T_{MAX_i} into set S_a. Sim_{S_1} randomly selects $r_j \in Z_N$, and inputs $([ID_{MAX_i}]_{pk_\Sigma})^{r_j}$ and $(T_{j,2})^{(N-r_j)}$ into $Sim_{S_1}^{\mathbf{SAD}}$ in Ref. [40] and gets V_j. Then, Sim_{S_1} partially decrypts V_j to $V'_j = \mathbf{PD1}_{SK_1}(V_j)$, and permutes (V_j, V'_j) using the permutation function π_i. The result is denoted as $(V_{\pi_i(j)}, V'_{\pi_i(j)})$. Sim_{S_1} sends $(V_{\pi_i(j)}, V'_{\pi_i(j)})$ and the intermediate encrypted data of $Sim_{S_1}^{\mathbf{MAX}_n}$ and $Sim_{S_1}^{\mathbf{SMD}}$ to $\mathcal{A}_{S_1}$. If $\mathcal{A}_{S_1}$ replies with $\perp$, then Sim_{S_1} outputs $\perp$. The IND-CPA security of PCTD ensures that $\mathcal{A}_{S_1}$'s view is indistinguishable from its view in the real world execution.

Sim_{S_2} simulates $\mathcal{A}_{S_2}$ as following. It randomly generates $\rho \in_R \{0,1\}$. If $\rho = 0$, it sets $A_{\pi_i(j)} = [0]_{pk_\Sigma}$; otherwise, it sets $A_{\pi_i(j)} = [1]_{pk_\Sigma}$. The generated ciphertexts $A_{\pi_i(j)}$ is computationally indistinguishable from the real world due to the IND-CPA security of PCTD. In both real and ideal world, the views of $\mathcal{A}_{S_2}$ are indistinguishable. □

Theorem 10. *The* **Top-K** *protocol is secure against the adversary $\mathcal{A}^*$ defined in the attack model.*

Proof. The adversary $\mathcal{A}^*$ is assumed to have the following abilities.

(1) $\mathcal{A}^*$ is assumed to be an outside adversary and eavesdrop all the communications to get the transmitted information. As $\mathcal{A}^*$ is assumed to be an outside adversary, $\mathcal{A}^*$ cannot get data owner A's private key sk_A, data user B's private key sk_B and B's authorization secret key sk_Σ. $\mathcal{A}^*$ also cannot get CP's partial strong key SK_1 and CSP's partial strong key SK_2.

If $\mathcal{A}^*$ eavesdrops the communication channel between the system user and CP, $\mathcal{A}^*$ could get the ciphertexts $(T_1, \cdots, T_n)$ that are transmitted at the beginning of the **Top-K** protocol, and the encrypted result $S_a = (T_{MAX_1}, \cdots, T_{MAX_k})$ that is transmitted at the end of the protocol, where $T_i = \langle [I_i]_{pk_\Sigma}, [ID_i]_{pk_{A_i}}, [K_i]_{pk_{A_i}} \rangle$. Since $(T_1, \cdots, T_n)$ and S_a are encrypted using the PCTD algorithm, the adversary $\mathcal{A}^*$ cannot recover (I_i, ID_i, K_i) for $1 \le i \le n$ and the plaintext underlying S_a due to the IND-CPA security of PCTD.

If $\mathcal{A}^*$ eavesdrops the communication channel between CP and CSP, $\mathcal{A}^*$ could get $(V_{\pi_i(j)}, V'_{\pi_i(j)})$ in line 8, and $A_{\pi_i(j)}$ in line 14 of the **Top-K** protocol, where $V_j = \mathbf{SAD}(([ID_{MAX_i}]_{pk_\Sigma})^{r_j}, (T_{j,2})^{(N-r_j)})$, $V'_j = \mathbf{PD1}_{SK_1}(V_j)$ and $A_{\pi_i(j)} = [0]_{pk_\Sigma}$ or $A_{\pi_i(j)} = [1]_{pk_\Sigma}$. In the **Top-K** protocol, these ciphertexts $(V_{\pi_i(j)}, V'_{\pi_i(j)}, A_{\pi_i(j)})$ are all encrypted using the public key pk_Σ. Since the adversary $\mathcal{A}^*$ does know the data owner B's authorization secret key sk_Σ and the CSP's partial strong key SK_2, $\mathcal{A}^*$ cannot recover the underlying plaintexts.

(2) $\mathcal{A}^*$ is assumed to compromise CP and get CP's partial strong key SK_1. But $\mathcal{A}^*$ cannot get CSP's partial strong key SK_2. $\mathcal{A}^*$ also cannot get data owner A's private key sk_A, data user B's private key sk_B and B's authorization secret key sk_Σ.

In line 6 of the **Top-K** protocol, $\mathcal{A}^*$ obtains

$$\begin{aligned} V_j &= \mathbf{SAD}(([ID_{MAX_i}]_{pk_\Sigma})^{r_j}, (T_{j,2})^{(N-r_j)}) \\ &= \mathbf{SAD}(([ID_{MAX_i}]_{pk_\Sigma})^{r_j}, [ID_j]_{pk_{A_j}}^{(N-r_j)}) \\ &= [r_j(ID_{MAX_i} - ID_j)]_{pk_\Sigma} \end{aligned}$$

$\mathcal{A}^*$ cannot recover $r_j(ID_{MAX_i} - ID_j)$ without the secret key sk_Σ. In line 14, $\mathcal{A}^*$ obtains $A_{\pi_i(j)}$ from CSP, where

$$A_{\pi_i(j)} = \begin{cases} [0]_{pk_\Sigma}, & if\ \beta_j = 0, \\ [1]_{pk_\Sigma}, & otherwise. \end{cases}$$

Since β_j and sk_Σ are unknown to CP, $\mathcal{A}^*$ cannot get the plaintext of $A_{\pi_i(j)}$ nor distinguish T_{MAX_i} comes from which element in $(T_1, \cdots, T_n)$.

(3) $\mathcal{A}^*$ is assumed to compromise CSP and get CSP's partial strong key SK_2. But $\mathcal{A}^*$ cannot get CP's partial strong key SK_1. $\mathcal{A}^*$ also cannot get data owner A's private key sk_A, data user B's private key sk_B and B's authorization secret key sk_Σ.

In line 8 of the **Top-K** protocol, $\mathcal{A}^*$ obtains $(V_{\pi_i(j)}, V'_{\pi_i(j)})$ transmitted by CP. Since $\mathcal{A}^*$ knows CSP's partial strong key SK_2, $\mathcal{A}^*$ decrypts

$$\begin{aligned} V''_{\pi_i(j)} &= \mathbf{PD2}_{SK_2}(V_{\pi_i(j)}, V'_{\pi_i(j)}) \\ &= r_{\pi_i(j)}(ID_{MAX_i} - ID_{\pi_i(j)}) \end{aligned}$$

Although $\mathcal{A}^*$ knows whether $r_{\pi_i(j)}(ID_{MAX_i} - ID_{\pi_i(j)})$ equals 0, the adversary $\mathcal{A}^*$ cannot distinguish $ID_{\pi_i(j)}$ comes from which element in $(ID_1, \cdots, ID_n)$. The reason is that CP utilizes a permutation function π_i to disrupt the order of $(V_1, \cdots, V_n)$ and $(V'_1, \cdots, V'_n)$ in line 8 of the **Top-K** protocol. Thus, $\mathcal{A}^*$ cannot distinguish T_{MAX_i} comes from which element in $(T_1, \cdots, T_n)$.

(4) $\mathcal{A}^*$ is assumed to be a set of collude malicious users $(B_1, \cdots, B_n)$ (except the challenge user B^*), and $\mathcal{A}^*$ gets their secret keys $(sk_{B_1}, \cdots, sk_{B_n})$. $\mathcal{A}^*$ wants to get the information that belongs to the challenge user B^*. Suppose the returned result is $S_a = \{T_{MAX_1}, \cdots, T_{MAX_k}\}$ that is transmitted at the end of the protocol, where $T_{MAX_i} = \langle [I_{MAX_i}]_{pk_{\Sigma^*}}, [ID_{MAX_i}]_{pk_{\Sigma^*}}, [K_{MAX_i}]_{pk_{\Sigma^*}} \rangle$ for $1 \le i \le k$, and pk_{Σ^*} is the authorize public key from data owner A to challenge user B^*. Since the user's secret keys are independently generated, the adversary $\mathcal{A}^*$ cannot utilize $(sk_{B_1}, \cdots, sk_{B_n})$ to deduce the challenge user B^*'s secret key sk_{B^*}. $\mathcal{A}^*$ also cannot get the authorization secret key sk_{Σ^*}. Thus, $\mathcal{A}^*$ cannot recover $(I_{MAX_i}, ID_{MAX_i}, K_{MAX_i})$ for $1 \le i \le k$.

According to the above analysis, the **Top-K** protocol is secure against the adversary $\mathcal{A}^*$ defined in the attack model. □